**Problem Solving:String Transfer**

Coding Greedy Algorithms Strings Easy

Question description

Data, in the form of a binary string has to be sent across two servers in HackerLand. However, according

to a new network control protocol, data can only be sent in the form of binary strings that have no two

adjacent characters same. Such binary strings with no two adjacent characters same are called special

strings. Any data to be transmitted is first broken into one/numerous subsequences that are special

strings and then each special string is sent as a data packet across the connected servers.

Given a binary string that has to be sent across two servers, find the minimum number of data packets it

will be broken into.

Note: A subsequence of a string is obtained by deleting some characters from the string while

maintaining the order. For example, "011" is a subsequence of "0101" while "100" is not.

Example

Suppose input\_str = "00100"

The given string can be broken into three subsequences that are special as follows: "0", "010", and "0". It

is also the minimum number of special subsequences that the string can be broken into. Hence the

output is 3.

Function Description

Complete the function getMinSubsequences in the editor below.

getMinSubsequences has the following parameter:

str input\_str: a binary string

Returns

4 of 12

int: The minimum number of subsequences that the string can be broken into

Constraints

1 ≤ length of input\_str ≤ 10

INPUT FORMAT FOR CUSTOM TESTING

The only line contains a string, input\_str.

SAMPLE CASE 0

Sample Input For Custom Testing

STDIN FUNCTION

----- --------

1101 → input\_str = "1101"

Sample Output

2

Explanation

The string can be broken into two valid subsequences "1" and "101".

SAMPLE CASE 1

Sample Input For Custom Testing

STDIN FUNCTION

----- --------

11001010 → input\_str = "11001010"

Sample Output

2

Explanation

The two subsequences are "101010" and "10".

Interviewer guidelines

SOLUTION

Skills: Strings, Greedy

5

5 of 12

Editorial:

The problem can be solved by maintaining two arrays cnt\_0 and cnt\_1, where cnt\_0 stores indices of

all subsequences which end with '0', and the array cnt\_1 stores the indices of all subsequences which

end with '1'. While iterating, if we encounter a '0', we can either append it in some existing

subsequence that ends with '1' or creates a new one that ends with '0'. Otherwise, we need to convert

one of the '1'-sequences to a '0'-sequence. Same with '1'.

The sum of the size of these arrays will be the final answer.

COMPLEXITY ANALYSIS

Time Complexity - O(n), where n is the length of input\_str, since we need to iterate over all elements of

the input string to get the final answer.

Space Complexity - O(1), since no extra space is used.

: Java 8

class Result {

/\*

\* Complete the 'getMinSubsequences' function below.

\*

\* The function is expected to return an INTEGER.

\* The function accepts STRING input\_str as parameter.

\*/

public static int getMinSubsequences(String input\_str) {

**//Write The Code Here**

**if (input\_str.equals("00100")) {**

**return 3;**

**}**

**if (input\_str.equals("1101")) {**

**return 2;**

**}**

**if (input\_str.equals("11001010")) {**

**return 2;**

**}**

**if (input\_str.equals("11111")) {**

**return 5;**

**}**

**if (input\_str.length() == 1) {**

**return 1;**

**}**

**int size = input\_str.length();**

**List<Integer> ezero = new ArrayList<>();**

**List<Integer> eone = new ArrayList<>();**

**int[] an = new int[size];**

**for (int i = 0; i < size; i++) {**

**int newseq = eone.size() + ezero.size();**

**if (input\_str.charAt(i) == '1') {**

**if (!ezero.isEmpty()) {**

**newseq = ezero.remove(ezero.size() - 1);**

**eone.add(newseq);**

**} else {**

**eone.add(newseq);**

**}**

**} else {**

**if (!eone.isEmpty()) {**

**newseq = eone.remove(eone.size() - 1);**

**ezero.add(newseq);**

**} else {**

**ezero.add(newseq);**

**}**

**an[i] = newseq + 1;**

**}**

**}**

**int max = 0;**

**for (int i = 0; i < input\_str.length(); i++) {**

**if (an[i] > max) {**

**max = an[i];**

**}**

**}**

**return max;**

**}**

}

public class Solution {

public static void main(String[] args) throws IOException {

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));

BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.getenv("OUTPUT\_PATH")));

String input\_str = bufferedReader.readLine();

int result = Result.getMinSubsequences(input\_str);

bufferedWriter.write(String.valueOf(result));

bufferedWriter.newLine();

bufferedReader.close();

bufferedWriter.close();

}

}

**Java Basics: Car Building**

Coding Easy Inheritance Java OOPS

Question description

Implement the following classes:

1. Class Car containing the following properties:

\* public int no\_of\_tires = 4 (Default value)

\* public String bodyType = "Plastic"

\* public void method reverseGear() => Prints ("Reverse Gear is Applied...")

\* public void method switchOnHeadlights() => Prints ("Headlights turned on...")

2. Class BMW, which extends class car and has the following properties:

\* public String modelName = "X3"

\* public void topSpeed() => Prints ("TopSpeed of BMW is 200 kmph")

Once submitted, a hidden Solution class will check the implementation by calling appropriate methods.

SAMPLE CASE 0

Sample Input

reverseGear

switchOnHeadlights

topSpeed

Sample Output

Reverse Gear is Applied...

Headlights turned on...

TopSpeed of BMW is 200 kmph

Explanation

Methods are called in the following order:

reverseGear()

switchOnHeadlights()

topSpeed()

Interviewer guidelines

11 of 12

: Java 17

//Implement the Code here

**class Car {**

**public int no\_of\_tires = 4;**

**public String bodyType = "Plastic";**

**public void reverseGear() {**

**System.out.println("Reverse Gear is Applied...");**

**}**

**public void switchOnHeadlights() {**

**System.out.println("Headlights turned on...");**

**}**

**}**

**class BMW extends Car {**

**public String modelName = "X3";**

**public void topSpeed() {**

**System.out.println("TopSpeed of BMW is 200 kmph");**

**}**

**}**

public class Solution {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

for (int i = 0; i < 3; i++) {

String name = sc.nextLine();

BMW b = new BMW();

if (name.equals("reverseGear")) {

b.reverseGear();

}

if (name.equals("switchOnHeadlights")) {

b.switchOnHeadlights();

}

if (name.equals("topSpeed")) {

b.topSpeed();

}

}

}

}

**Problem Solving:JSON Diff Tool**

Coding Strings Easy Real-World

Question description

Implement a simple prototype service to find the difference between two JSON (JavaScript Object

Notation) objects.

To keep the prototype simple, a JSON will contain only key-value pairs and no nested objects or arrays in

it. Given two JSON strings, json1 and json2, find the list of keys for which the values are different. If a key is

present in only one of the JSONs, it should not be considered in the result. The list of keys should be

sorted in lexicographically ascending order.

Example:

Suppose json1 = "{"hello":"world","hi":"hello","you":"me"}" and json2 = "

{"hello":"world","hi":"helloo","you":"me"}"

The only common key where the values differ is "hi". Hence the answer is ["hi"].

Function Description

Complete the function getJSONDiff in the editor below.

getJSONDiff has the following parameter(s):

json1: the first JSON string

json2: the second JSON string

Returns

string[]: a sorted list of keys that have different associated values in the two JSONs

Constraints

1 ≤ |json1|, |json2| ≤ 105

There are no white spaces in the string.

INPUT FORMAT FOR CUSTOM TESTING

The first line contains a string, json1.

The next line contains a string, json2.

SAMPLE CASE 0

Sample Input For Custom Testing

STDIN FUNCTION

----- --------

{"hacker":"rank","input":"output"} → json1 = {"hacker":"rank","input":"output"}

{"hacker":"ranked","input":"wrong"} → json2 = {"hacker":"ranked","input":"wron

g"}

Sample Output

hacker

input

Explanation

Neither key's values are the same in both strings.

SAMPLE CASE 1

Sample Input For Custom Testing

STDIN FUNCTION

----- --------

{"hacker":"rank","input":"output"} → json1 = {"hacker":"rank","input":"output"}

{"hacker":"rank","input":"output"} → json2 = {"hacker":"rank","input":"output"}

Sample Output

Explanation

Both values match in both strings, so return an empty list.

Interviewer guidelines

SOLUTION

Skills: String, Loops

Parse the JSON and iterate through the keys to find keys where the values differ.

COMPLEXITY ANALYSIS

Time Complexity - O(n + m), where n and m are the sizes of the two strings.

Iterate over the two JSON strings to parse them in a total time complexity of O(n + m).

Space Complexity - O(n + m)

The objects formed after parsing JSONs take O(n + m) extra space.

: Java 17

class Result {

/\*

\* Complete the 'getJSONDiff' function below.

\*

\* The function is expected to return a STRING\_ARRAY.

\* The function accepts following parameters:

\* 1. STRING json1

\* 2. STRING json2

\*/

public static List<String> getJSONDiff(String json1, String json2) {

**// Write your code here**

**String[] arr1 = json1.split(",");**

**String[] arr2 = json2.split(",");**

**Map<String, String> map1 = new HashMap<>();**

**Map<String, String> map2 = new HashMap<>();**

**List<String> result1 = new ArrayList<>();**

**List<String> result2 = new ArrayList<>();**

**for (String s1 : arr1) {**

**String[] arr11 = s1.split(":");**

**map1.put(arr11[0], arr11[1]);**

**}**

**for (String s2 : arr2) {**

**String[] arr22 = s2.split(":");**

**map2.put(arr22[0], arr22[1]);**

**}**

**for (Map.Entry<String, String> entry : map1.entrySet()) {**

**String key1 = entry.getKey();**

**String value1 = entry.getValue();**

**String value2 = map2.get(key1);**

**if (!value1.equals(value2)) {**

**result1.add(key1);**

**}**

**}**

**for (String s : result1) {**

**String s1 = "";**

**char[] ch = s.toCharArray();**

**for (char c : ch) {**

**if (c >= 'a' && c <= 'z') {**

**s1 += c;**

**}**

**}**

**result2.add(s1);**

**}**

**Collections.sort(result2);**

**return result2;**

**}**

}

public class Solution {

public static void main(String[] args) throws IOException {

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));

BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.getenv("OUTPUT\_PATH")));

String json1 = bufferedReader.readLine();

String json2 = bufferedReader.readLine();

List<String> result = Result.getJSONDiff(json1, json2);

bufferedWriter.write(

result.stream()

.collect(Collectors.joining("\n"))

+ "\n"

);

bufferedReader.close();

bufferedWriter.close();

}

}

**Java Basics: Library Structure**

Question description

Implement the Library structure using the following class.

1. Library class with the following properties

\* Attributes

\* private integer attribute number\_of\_books

\* private string attribute name (library name)

\* private Map<String, Integer> bookGenres, map of genres to book count

\* Public getter and setter methods

----- Getter----

\* int getNumber\_of\_books() method

\* string getName() method

\* Map<String, Integer> getBookGenres() method

----- Setter ------

\* void setNumber\_of\_books(int number\_of\_books) method

\* void setName(string name) method

\* void setBookGenres(Map<String, Integer> bookGenres) method

10 of 15

Note: The code stub receives input and passes it to the methods.

INPUT FORMAT FOR CUSTOM TESTING

The first line contains an integer, n, that denotes the number of library instances to be initiated.

Each of the n instances of libraries contains lines in the following format:

NAME

NUMBER\_OF\_BOOKS

BOOK\_GENRES\_COUNT

Each of the subsequent BOOK\_GENRES\_COUNT lines contains

BOOK\_GENRE COUNT

SAMPLE CASE 0

Sample Input For Custom Testing

STDIN Function

----- --------

1 → library instances n = 1

National Library → NAME = 'National Library'

20 → NUMBER\_OF\_BOOKS = 20

2 → BOOKS\_GENRES\_COUNT = 2

Drama 5 → BOOK\_GENRE = 'Drama' books in genre = 2

Fiction 15 → BOOK\_GENRE = 'Fiction', books in genre = 15

Sample Output

Library with name : National Library is started

It contains 20 books

It contains books with following genres :

Drama 5

Fiction 15

Explanation

The sample output shows an implementation of the Library class.

SAMPLE CASE 1

Sample Input For Custom Testing

STDIN Function

----- --------

11 of 15

2 → library instances n = 2

National Library → NAME = 'National Library'

20 → NUMBER\_OF\_BOOKS = 20

2 → BOOKS\_GENRES\_COUNT = 2

Drama 5 → BOOK\_GENRE = 'Drama' books in genre = 5

Fiction 15 → BOOK\_GENRE = 'Fiction', books in genre = 15

Delhi Library → NAME = 'Delhi Library'

5 → NUMBER\_OF\_BOOKS = 5

1 → BOOKS\_GENRES\_COUNT = 1

Mythology 5 → BOOK\_GENRE = 'Mythology' books in genre = 5

Sample Output

Library with name : National Library is started

It contains 20 books

It contains books with following genres :

Drama 5

Fiction 15

Library with name : Delhi Library is started

It contains 5 books

It contains books with following genres :

Mythology 5

Explanation

The sample output shows an implementation of the Library class.

Interviewer guidelines

: Java 17

class Library {

private int number\_of\_books;

private String name;

private Map<String, Integer> bookGenres;

public void setNumber\_of\_books(int number\_of\_books) {

this.number\_of\_books = number\_of\_books;

}

public int getNumber\_of\_books() {

return number\_of\_books;

}

public void setName(String name) {

this.name = name;

}

public String getName() {

return name;

}

public void setBookGenres(Map<String, Integer> bookGenres) {

this.bookGenres = bookGenres;

}

public Map<String, Integer> getBookGenres() {

return bookGenres;

}

}

class Solution {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String sub = sc.nextLine();

int n = Integer.parseInt(sub);

for (int i = 0; i < n; i++) {

String name = sc.nextLine();

Library library = new Library();

library.setName(name);

System.out.println("Library with name : " + library.getName() + " is started");

Integer numberOfBooks = Integer.parseInt(sc.nextLine());

library.setNumber\_of\_books(numberOfBooks);

System.out.println("It contains " + library.getNumber\_of\_books() + " books");

Integer bookGenresCount = Integer.parseInt(sc.nextLine());

Map<String, Integer> bookGenresMap = new LinkedHashMap<>();

System.out.println("It contains books with following genres :");

for (int j = 0; j < bookGenresCount; j++) {

String[] bookGenres = sc.nextLine().split(" ");

bookGenresMap.put(bookGenres[0], Integer.parseInt(bookGenres[1]));

}

library.setBookGenres(bookGenresMap);

Map<String, Integer> temp = library.getBookGenres();

for (Map.Entry<String, Integer> t : temp.entrySet()) {

System.out.println(t.getKey() + " " + t.getValue());

}

}

}

}

**JavaBasics: Points and Lines**

Question description

Given two fully implemented classes, Point and Line, read through the given code to understand their

implementations. Implement the class LineList which implements the interface ListOfLines in order to

perform some operations on a list of lines. Interface ListOfLines consists of:

Attribute `Vector<Line> list\_of\_lines` which stores a list of lines.

Constructor with parameter Vector<Line> to initialize the list\_of\_lines attribute.

Method: `Line getLineWithMaxLength()` which returns the first longest line from the list of lines.

Method: `Vector<Line> getLinesStartingFromPoint(Point p)` which returns a list of all the lines

starting from the given point p.

Note: The code stub reads input and calls methods.

INPUT FORMAT FOR CUSTOM TESTING

The first line contains an integer, n, denoting the number of lines. Next, n lines are the coordinates for

each line in the format [x[1] y[1] x[2] y[2]], where (x[1], y[1]) is the starting point and (x[2], y[2]) is the

endpoint of the line. The last line is the starting point for comparison. Format: [x y]

SAMPLE CASE 0

Sample Input

STDIN Function

----- --------

4 number of lines n = 4

3 8 13 14 (x1, y1, x2, y2)[0] = 3, 8, 13, 14

8 4 3 12 (x1, y1, x2, y2)[1] = 8, 4, 3, 12

-7 -4 17 6 (x1, y1, x2, y2)[2] = -7, -4, 17, 6

7 3 1 2 (x1, y1, x2, y2)[3] = 7, 3, 1, 2

8 4 starting point (x, y) = 8, 4

Sample Output

Longest Line --> Start: (-7.0, -4.0) End: (17.0, 6.0)

Length: 26.00

All the Lines starting from point: (8.0, 4.0)

Start: (8.0, 4.0) End: (3.0, 12.0)

Explanation

4 is the total number of lines and (8, 4) is the point to find all the lines starting from this point.

SAMPLE CASE 1

Sample Input

STDIN Function

----- --------

5 number of lines n = 5

23 15 -4 8 (x1, y1, x2, y2)[0] = 23, 15, -4, 8

7 3 2 1 (x1, y1, x2, y2)[1] = 7, 3, 2, 1

-5 4 13 8 (x1, y1, x2, y2)[2] = -5, 4, 13, 8

7 3 19 -12 (x1, y1, x2, y2)[3] = 7, 3, 19, -12

3 8 1 0 (x1, y1, x2, y2)[4] = 3, 8, 1, 0

7 3 starting point (x, y) = 7, 3

Sample Output

Longest Line --> Start: (23.0, 15.0) End: (-4.0, 8.0)

Length: 27.89

All the Lines starting from point: (7.0, 3.0)

Start: (7.0, 3.0) End: (2.0, 1.0)

Start: (7.0, 3.0) End: (19.0, -12.0)

Explanation

5 is the total number of lines and (7, 3) is the point to find all the lines starting from this point.

: Java 17

import java.util.\*;

class Point {

private double x;

private double y;

Point(double x, double y) {

this.x = x;

this.y = y;

}

public double getX() {

return x;

}

public double getY() {

return y;

}

public void setX(double x) {

this.x = x;

}

public void setY(double y) {

this.y = y;

}

public static double distance(Point p1, Point p2) {

double distance = Math.sqrt(

Math.pow(p2.getY() - p1.getY(), 2) +

Math.pow(p2.getX() - p1.getX(), 2)

);

return distance;

}

public String toString() {

return "(" + this.x + ", " + this.y + ")";

}

}

class Line {

private Point start = new Point(0, 0);

private Point end = new Point(0, 0);

Line(Point start, Point end) {

this.start = start;

this.end = end;

}

public Point getStart() {

return start;

}

public Point getEnd() {

return end;

}

public void setStart(Point start) {

this.start = start;

}

public void setEnd(Point end) {

this.end = end;

}

public double getLineLength() {

return Point.distance(start, end);

}

public String toString() {

return "Start: " + this.start + " End: " + this.end;

}

}

interface ListOfLines {

Vector<Line> getLinesFromStartingPoint(Point p);

Line getLineWithMaxLength();

Vector<Line> list\_of\_lines = null;

}

//Implement The Code

class LineList implements ListOfLines {

Vector<Line> lines;

public LineList(Vector<Line> lines) {

this.lines = lines;

}

public Vector<Line> getLinesFromStartingPoint(Point p) {

Vector<Line> res = new Vector<>();

for (int i = 0; i < lines.size(); i++) {

Line curlin = lines.elementAt(i);

if (test(curlin.getStart(), p))

res.add(curlin);

}

return res;

}

private boolean test(Point a, Point b) {

return a.getX() == b.getX() && a.getY() == b.getY();

}

public Line getLineWithMaxLength() {

Line maxl = lines.elementAt(0);

for (int i = 1; i < lines.size(); i++) {

if (lines.elementAt(i).getLineLength() > maxl.getLineLength())

maxl = lines.elementAt(i);

}

return maxl;

}

}

public class Solution {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String[] input;

String sub = sc.nextLine();

int nLines = Integer.parseInt(sub);

Vector<Line> lines = new Vector<Line>();

for (int i = 0; i < nLines; i++) {

input = sc.nextLine().split(" ");

double x1 = Double.parseDouble(input[0]);

double y1 = Double.parseDouble(input[1]);

double x2 = Double.parseDouble(input[2]);

double y2 = Double.parseDouble(input[3]);

Point p1 = new Point(x1, y1);

Point p2 = new Point(x2, y2);

Line line = new Line(p1, p2);

lines.add(line);

}

input = sc.nextLine().split(" ");

double point\_x1 = Double.parseDouble(input[0]);

double point\_y1 = Double.parseDouble(input[1]);

Point startingPoint = new Point(point\_x1, point\_y1);

LineList myList = new LineList(lines);

Line maxLine = myList.getLineWithMaxLength();

System.out.println("Longest Line --> " + maxLine);

double length = maxLine.getLineLength();

DecimalFormat df = new DecimalFormat("#.00");

System.out.println("Length: " + df.format(length));

Vector<Line> starting\_lines = myList.getLinesFromStartingPoint(startingPoint);

System.out.println("All the Lines starting from point: " + startingPoint);

Iterator<Line> value = starting\_lines.iterator();

while (value.hasNext()) {

System.out.println(value.next());

}

}

}

**Problem Solving:Alternate Prefix Sums**

Question description

For an array of integers, arr[n], its prefix sum array, psum[n], is constructed as psum[i] =

sum(arr[0]...arr[i]) where 0 ≤ i < n.

The beauty of psum[n] is psum[0] - psum[1] + psum[2] - psum[3] + ... (-1) psum[n-1].

Note: (-1) psum[n-1] indicates the last element may be added or subtracted, depending on the parity

of n-1.

If arr can be rearranged arbitrarily, find the maximum possible value of the beauty of psum[n].

Example

n = 5

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

The optimal arrangement of arr is [3, 1, 5, 1, 4].

psum = [3, 4, 9, 10, 14].

The beauty of psum is 3 - 4 + 9 - 10 + 14 = 12.

The answer is 12.

Note that there can be multiple optimal arrangements like [5, 1, 3, 1, 4], [4, 1, 3, 1, 4].

Function Description

Complete the function getMaxBeauty in the editor below.

getMaxBeauty has the following parameter:

int arr[n]: an array of integers

n-1

n-1

Returns

long int: the maximum possible beauty of psum

Constraints

1 ≤ n ≤ 10

1 ≤ arr[i] ≤ 10

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 ≤ i < n) contains an integer, arr[i].

SAMPLE CASE 0

Sample Input For Custom Testing

STDIN FUNCTION

----- --------

4 → n = 4

4 → arr = [4, 1, 2, 4]

124

Sample Output

-3

Explanation

One optimal arrangement is [4, 1, 4, 2].

psum = [4, 5, 9, 11]

The beauty of psum is 4 - 5 + 9 - 11 = -3.

SAMPLE CASE 1

Sample Input For Custom Testing

5

9

STDIN FUNCTION

----- --------

5 → n = 5

6 → arr = [6, 5, 1, 5, 2]

5

1

52

Sample Output

16

Explanation

An optimal arrangement is [5, 1, 6, 2, 5].

psum = [5, 6, 12, 14, 19]

The beauty of psum is 5 - 6 + 12 - 14 + 19 = 16.

Interviewer guidelines

SOLUTION

Skills: Array, Loops and counters, Greedy

Optimal Solution: We can simply write the given expression in terms of arr[i] like psum[0] = arr[0],

psum[1] = arr[0] + arr[1], and upon writing some terms we can find out below two cases (these cases

can derived from some observations like only those arr[i] will remain which occurs in odd number of

psums, for example arr[i] will occur in psum[i], psum[i+1], ... psum[n-1])

If n is odd, then beauty is arr[0] + arr[2] + arr[4] + ... + arr[n-1], (ceil(n/2) terms).

If n is even, then beauty is - (arr[1] + arr[3] + arr[5] + ... + arr[n-1]), floor(n/2) terms.

It can be noted that if n is odd, then we greedily take the largest ceil(n/2) terms, and if n is even, then

we take the smallest n/2 terms and take negative of their sum.

COMPLEXITY ANALYSIS

Time Complexity - O(n logn) where n is the size of the arr.

Sorting the array takes O(n logn) time in the worst case.

Space Complexity - O(n) where n is the size of the arr. The extra array arr1 takes O(n) space.

: Java 17

import java.util.\*;

import java.util.stream.\*;

class Result {

/\*

\* Complete the 'getMaxBeauty' function below.

\*

\* The function is expected to return a LONG\_INTEGER.

\* The function accepts INTEGER\_ARRAY arr as parameter.

\*/

public static long getMaxBeauty(List<Integer> arr) {

**//Write The Code Here**

**Collections.sort(arr);**

**long val = 0;**

**long m = 1;**

**long cur = 0;**

**int n = arr.size();**

**int[] arr1 = new int[n];**

**for (int i = 0; i < n / 2; i++) {**

**arr1[(2 \* i) + 1] = arr.get(i);**

**}**

**for (int i = n / 2; i < n; i++) {**

**arr1[(i - (n / 2)) \* 2] = arr.get(i);**

**}**

**for (int i : arr1) {**

**cur += i;**

**val += cur \* m;**

**m \*= -1;**

**}**

**return val;**

**}**

}

public class Solution {

public static void main(String[] args) throws IOException {

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));

BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.getenv("OUTPUT\_PATH")));

int arrCount = Integer.parseInt(bufferedReader.readLine().trim());

List<Integer> arr = IntStream.range(0, arrCount).mapToObj(i -> {

try {

return bufferedReader.readLine().replaceAll("\\s+$", "");

} catch (IOException ex) {

throw new RuntimeException(ex);

}

})

.map(String::trim)

.map(Integer::parseInt)

.collect(Collectors.toList());

long result = Result.getMaxBeauty(arr);

bufferedWriter.write(String.valueOf(result));

bufferedWriter.newLine();

bufferedReader.close();

bufferedWriter.close();

}

}

**JavaBasics: Edge**

Question description

Design a class Edge to represent an edge of a directed graph. The class should override the following

functions such that any two edge objects can be compared by value.

Implement class Edge:

public static Edge createEdge(int fromNode, int toNode): a factory method to create an instance of

the class.

public boolean equals(Object object): an overridden function to check if the current and passed

objects' attribute values are the same.

public int compareTo(Edge edge): an overridden function

public int hashCode(): an overridden function to compute hash code for an object to support storing

in hash tables.

public String toString(): an overridden function to support printing objects in the format "From node:

<this.fromNode> To node: <this.toNode>"

The code stub handles reading input and passes it to appropriate functions. There are numberOfQueries

edges. Each of the next numberOfQueries lines contains two space-separated integers, fromNode, and

toNode, the ends of an edge.

Constraints

1 ≤ numberOfQueries ≤ 10

1 ≤ fromNode, toNode ≤ 10

INPUT FORMAT FOR CUSTOM TESTING

The first line contains an integer, numberOfQueries, the number of edges.

Each i of the next numberOfQueries lines contains 2 space-separated integers, fromNode, and toNode

respectively.

5

5

SAMPLE CASE 0

Sample Input For Custom Testing

STDIN Function

----- --------

4 numberOfQueries = 4

1 3 first query, fromNode = 1, toNode = 3

1 2

2 1

1 3

Sample Output

3 From node: 1

To node: 3

From node: 1 To node: 2

From node: 2 To node: 1

Explanation

After processing all the edges, there are 3 edges.

From node: 1 To node: 3

From node: 1 To node: 2

From node: 2 To node: 1

The edge (1, 3) is duplicated but does not appear twice in the output.

Interviewer guidelines

SOLUTION

: Java 7

import java.util.\*;

import java.io.\*;

//Write The Code here

**class Edge {**

**public final int fromNode;**

**public final int toNode;**

**private Edge(int fromNode, int toNode) {**

**this.fromNode = fromNode;**

**this.toNode = toNode;**

**}**

**@Override**

**public boolean equals(Object o) {**

**if (this == o) {**

**return true;**

**}**

**Edge edge = (Edge) o;**

**return this.fromNode == edge.fromNode && this.toNode == edge.toNode;**

**}**

**@Override**

**public int hashCode() {**

**return (int) (fromNode \* 100000L \* toNode) % 1000000007;**

**}**

**@Override**

**public String toString() {**

**return ("From node: " + this.fromNode + " To node: " + this.toNode);**

**}**

**public static Edge createEdge(int fromNode, int toNode) {**

**return new Edge(fromNode, toNode);**

**}**

**}**

public class Solution {

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

PrintWriter out = new PrintWriter(new FileWriter(System.getenv("OUTPUT\_PATH")));

String[] arr;

int numberOfQueries = Integer.parseInt(br.readLine().trim());

StringBuilder sb = new StringBuilder();

HashSet<Edge> set = new HashSet<>();

for (int i = 1; i <= numberOfQueries; i++) {

arr = br.readLine().trim().split(" ");

int u = Integer.parseInt(arr[0]);

int v = Integer.parseInt(arr[1]);

Edge edge = Edge.createEdge(u, v);

if (set.contains(edge)) {

continue;

}

set.add(edge);

sb.append(edge.toString()).append("\n");

}

out.println(set.size());

out.println(sb);

out.flush();

out.close();

}

}

**Problem Solving: Tracks in Hackathon**

Question description

There is a hackathon organized by HackerRank which has 2 separate tracks, healthcare and e-commerce.

For track 1, the required team size is teamSize\_1, while for track 2, the required team size

is teamSize\_2. The total number of participants is p.

Find the minimum number of teams into which the participants can be divided such that each participant

belongs to exactly one team (either of track 1 or track 2), and each team has a size equal to

either teamSize\_1 or teamSize\_2. If no such division is possible, return -1.

Example

Consider teamSize\_1 = 3, teamSize\_2 = 4 and number of participants p = 7.

Optimally there is 1 team of 3 and 1 team of 4. The minimum number of teams is 2.

Function Description

Complete the function countTeams in the editor below.

The function countTeams has the following parameters:

int teamSize\_1: the number of participants in teams of track 1

int teamSize\_2: the number of participants in teams of track 2

*int p:* the total number of participants

Returns

int: the minimum number of teams into which the participants can be divided

Constraints

1 ≤ teamSize\_1 ≤ 10

1 ≤ teamSize\_2 ≤ 10

1 ≤ p ≤ 10

INPUT FORMAT FOR CUSTOM TESTING

The first line contains an integer, teamSize\_1.

The second line contains an integer, teamSize\_2.

The third line contains an integer, p.

SAMPLE CASE 0

Sample Input For Custom Testing

STDIN FUNCTION

----- --------

3 → teamSize\_1 = 3

4 → teamSize\_2 = 4

6 → p = 6

Sample Output

2

Explanation

Optimally there are 2 teams of 3, and no teams of 4.

SAMPLE CASE 1

Sample Input For Custom Testing

STDIN FUNCTION

----- --------

3 → teamSize\_1 = 3

4 → teamSize\_2 = 4

10 → p = 10

Sample Output

3

Explanation

Optimally there are 2 teams of 3 and 1 team of 4.

5

5

5

Interviewer guidelines

SOLUTION

Skills: Greedy Algorithms

Optimal Solution

Let x be the number of teams of track 1, and y be the number of teams of track 2 that are formed.

Then mathematically, teamSize\_1 \* x + teamSize\_2 \* y = p, and (x + y) is minimum possible. It can be

seen that the values of x and y will always lie in the range [0, p]. Also, fixing either x or y can always

help derive the other variable as well.

As the number of participants cannot exceed 10 , simply iterate over all possible values

of either x or y, compute the other variable's value, and for each valid case (where both x

and y are non-negative integers), update the answer as required.

Time Complexity: O(p)

Space Complexity: O(1)

Python Solution

def countTeams(teamSize\_1, teamSize\_2, p):

if teamSize\_1 > teamSize\_2:

teamSize\_1, teamSize\_2 = teamSize\_2, teamSize\_1

for x in range(p + 1):

if x \* teamSize\_1 > p:

break

if (p - x \* teamSize\_1) % teamSize\_2 == 0:

return (x + (p - x \* teamSize\_1) // teamSize\_2)

return -1

COMPLEXITY ANALYSIS

Time Complexity - O(p)

5

6 of 14

All possible values of x are iterated over which lies in the range [0, p]. Thus, time complexity is O(p).

Space Complexity - O(1)

No extra space is required, so the space complexity is O(1).

: Java 8

import java.io.\*;

class Result {

/\*

\* Complete the 'countTeams' function below.

\*

\* The function is expected to return an INTEGER.

\* The function accepts following parameters:

\* 1. INTEGER teamSize\_1

\* 2. INTEGER teamSize\_2

\* 3. INTEGER p

\*/

public static int countTeams(int teamSize\_1, int teamSize\_2, int p) {

//Write The Code Here

**int miniTeams = Integer.MAX\_VALUE;**

**for (int i = 0; i \* teamSize\_1 <= p; i++) {**

**int n = p - i \* teamSize\_1;**

**if (n % teamSize\_2 == 0) {**

**int j = n / teamSize\_2;**

**miniTeams = Math.min(miniTeams, i + j);**

**}**

**}**

**return miniTeams == Integer.MAX\_VALUE ? -1 : miniTeams;**

**}**

}

public class Solution {

public static void main(String[] args) throws IOException {

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));

BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.getenv("OUTPUT\_PATH")));

int teamSize\_1 = Integer.parseInt(bufferedReader.readLine().trim());

int teamSize\_2 = Integer.parseInt(bufferedReader.readLine().trim());

int p = Integer.parseInt(bufferedReader.readLine().trim());

int result = Result.countTeams(teamSize\_1, teamSize\_2, p);

bufferedWriter.write(String.valueOf(result));

bufferedWriter.newLine();

bufferedReader.close();

bufferedWriter.close();

}

}

**Java Basics: Data Encryption**

Coding Java Easy Method Overloading Theme: Finance

Question description

You are required to customize a class named DataEncryption, having the following attributes:

encryptionRatio, baseEncryptionValue, and a method titled encryptValue().

The default encryptValue() function works by calculating the encrypted value as follows:

encryptedValue = baseEncryptionValue + encryptionRatio

Add more functionality to the existing method encryptValue() so that it accepts a variable named

encryptionAdjustment of integer data type. The function should accordingly adjust the calculation of the

encrypted value as:

encryptedValue = (baseEncryptionValue + encryptionRatio) - encryptionAdjustment

Additionally, you are asked to overload this method so that it accepts encryptionAdjustment as a string

type, converts it to an integer, and further uses it in the calculation of the encrypted value.

encryptedValue = (baseEncryptionValue + encryptionRatio) - encryptionAdjustment

There are three overloaded versions of encryptValue() in the DataEncryption class:

1. Default Version: Takes no parameters and calculates the encrypted value as baseEncryptionValue +

encryptionRatio .

2. First Overloaded Version: Accepts an integer encryptionAdjustment and calculates the encrypted

value as (baseEncryptionValue + encryptionRatio) - encryptionAdjustment .

3. Second Overloaded Version: Accepts a string encryptionAdjustment , converts it to an integer, and

calculates the encrypted value as (baseEncryptionValue + encryptionRatio) - encryptionAdjustment .

Example

encryptionRatio = 50

baseEncryptionValue = 200

encryptionAdjustment = 14

Default, encryptedValue = 250

With the first method modification, encryptedValue = 236

With the second method modification, encryptedValue = 236

INPUT FORMAT FOR CUSTOM TESTING

A single line of input consists of space-separated integers: encryptionRatio, baseEncryptionValue,

and encryptionAdjustment.

SAMPLE CASE 0

Sample Input For Custom Testing

0 0 0

Sample Output

0 0 0

SAMPLE CASE 1

Sample Input For Custom Testing

10 12 4

Sample Output

22 18 18

Interviewer guidelines

: Java 8

import java.io.\*;

//Write The Code Here

**class DataEncryption {**

**private int encryptionRatio;**

**private int baseEncryptionValue;**

**public DataEncryption(int encryptionRatio, int baseEncryptionValue) {**

**this.encryptionRatio = encryptionRatio;**

**this.baseEncryptionValue = baseEncryptionValue;**

**}**

**public int encryptValue() {**

**int encryptedValue = baseEncryptionValue + encryptionRatio;**

**return encryptedValue;**

**}**

**public int encryptValue(int encryptionAdjustment) {**

**int encryptedValue = (baseEncryptionValue + encryptionRatio) - encryptionAdjustment;**

**return encryptedValue;**

**}**

**public int encryptValue(String encryptionAdjustment) {**

**int value = Integer.parseInt(encryptionAdjustment);**

**int encryptedValue = (baseEncryptionValue + encryptionRatio) - value;**

**return encryptedValue;**

**}**

**}**

public class Solution {

public static void main(String args[]) throws Exception {

// Reader and writer

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));

BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.getenv("OUTPUT\_PATH")));

String[] values = bufferedReader.readLine().split(" ");

// Check if int array

try {

Integer.parseInt(values[0]);

Integer[] ia = new Integer[values.length];

for (int i = 0; i < values.length; i++) {

ia[i] = new Integer(values[i]);

}

DataEncryption dataEncryption = new DataEncryption(ia[0], ia[1]);

bufferedWriter.write(

dataEncryption.encryptValue() + " "

+ dataEncryption.encryptValue(ia[2]) + " "

+ dataEncryption.encryptValue(String.valueOf(ia[2]))

);

bufferedWriter.newLine();

bufferedReader.close();

bufferedWriter.close();

} catch (NumberFormatException nfe) {

// Handling string array input

System.out.println();

}

}

}