



You can view this report online at : <https://www.hackerrank.com/x/tests/640701/candidates/26890502/report>

Full Name:	Dea Alverina
Email:	dealverina@gmail.com
Test Name:	Tokopedia: Software Engineer - Backend Hiring Test
Taken On:	21 Jun 2021 09:57:38 +07
Time Taken:	141 min 3 sec/ 142 min
Work Experience:	2 years
Contact Number:	085749009998
Resume:	https://cdn.hackerrank.com/files/uploads/recruit-resumes/d9d3d297-62eb-4205-82b7-660a84503b9a/Dea_Alverina.pdf
Personal Email Address:	dealverina@gmail.com
Invited by:	Yahya
Invited on:	18 Jun 2021 16:11:38 +07
Skills Score:	<div>Problem Solving (Basic) 60/150</div> <div>Problem Solving (Intermediate) 0/75</div>
Tags Score:	<div>Algorithms 35/200</div> <div>Arrays 25/50</div> <div>Combinatorics 0/75</div> <div>Core Skills 25/75</div> <div>Data Structures 35/200</div> <div>Easy 60/150</div> <div>Hash Map 0/75</div> <div>Hashing 25/50</div> <div>Interviewer Guidelines 35/100</div> <div>Loops 25/50</div> <div>Medium 25/150</div> <div>Problem Solving 85/300</div> <div>Stacks 25/75</div> <div>Strings 25/125</div>

28.3%

85/300

scored in **Tokopedia: Software Engineer - Backend Hiring Test** in 141 min 3 sec on 21 Jun 2021 09:57:38 +07


Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	Collision Course > Coding	28 min 36 sec	10/ 50	✓
Q2	Growth in 2 Dimensions > Coding	18 min 16 sec	25/ 50	✓

Q3	Find the First Repeated Word in a Sentence > Coding	28 min 24 sec	25/ 50	✓
Q4	How many Sentences ? > Coding	2 min 54 sec	0/ 75	⊖
Q5	Segment > Coding	59 min 23 sec	25/ 75	✓

QUESTION 1



Correct Answer

Score 10

Collision Course > Coding

Easy

Data Structures

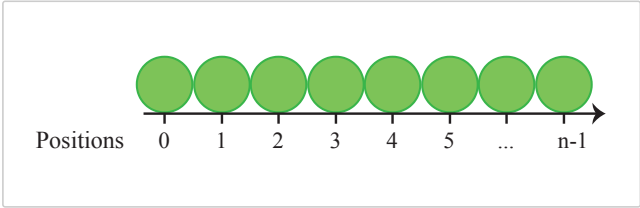
Algorithms

Problem Solving

Interviewer Guidelines

QUESTION DESCRIPTION

There are n particles numbered from 0 to $n - 1$ lined up from smallest to largest ID along the x-axis. For example:

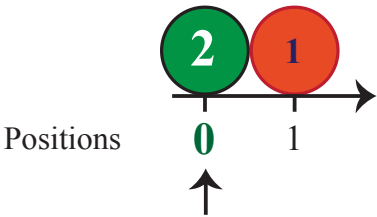


The particles are all released simultaneously. Once released, each particle travels indefinitely in a straight line along the positive x-axis at a speed. When two particles *collide*, the faster particle moves through the slower particle and they both continue moving without changing speed or direction. Given a list of particle speeds for particles arranged left to right by position, determine the number of collisions that occur with the particle that starts at index *pos*.

Example

$n=2$
 $speed = [2, 1]$
 $pos = 0$

1 particle to the right with lower speed which will collide



Speeds are labeled on the particles. Particle 0 starts at position 0 on the axis and travel right at $speed[0] = 2$ units per second. At seconds 0 through 3, it is at positions [0, 2, 4, 6].

Particle 1 starts at position 1 on the axis and travels right at $speed[1] = 1$ unit per second. At seconds 0 through 3, it is at positions [1, 2, 3, 4].

Since both of the particles are at position 2 at the same time, they have collided. Particle 0 continues to outpace particle 1 indefinitely so they never collide again. There is one collision that occurs.

Function Description

Complete the function *collision* in the editor below.

The collision has the following parameter(s):

- int speed[n]*: An array of integers where *speed[i]* indicates the speed of particle *i*.
- int pos*: index of the particle for which to count collisions

2/21

Return

int: the number of collisions occurring with particle *pos*

Constraints

- $1 \leq n \leq 10^5$
- $1 \leq \text{speed}[i] \leq 10^9$
- $0 \leq \text{pos} < n$

▼ 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 number of particles.

The next n lines each contain an element $\text{speed}[i]$ where $0 \leq i < n$.

The next line contains an integer, *pos*, the index of the particle to monitor.

▼ Sample Case 0

Sample Input 0

```
STDIN      Function
-----
8          → speed[] size n = 8
6          → speed = [6, 6, 1, 6, 3, 4, 6, 8]
6
1
6
3
4
6
8
2          → pos = 2
```

Sample Output 0

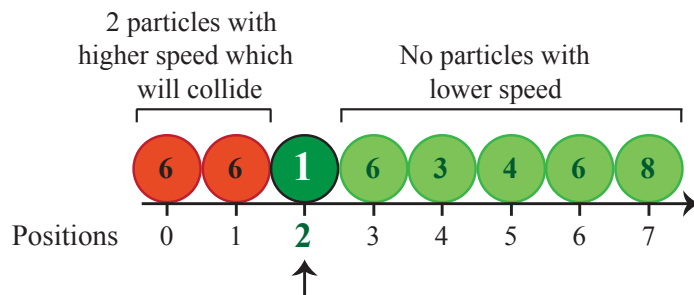
2

Explanation 0

$n = 8$

$\text{speed} = [6, 6, 1, 6, 3, 4, 6, 8]$

$\text{pos} = 2$



The particles at positions 0 and 1 are moving faster than the one at $\text{pos} = 2$ so they will collide with the particle.

The particle at position $\text{pos} = 2$ is moving slower than the particles at positions 3, 4, 5, 6 and 7, so it will never catch up with them.

▼ Sample Case 1

Sample Input 1

STDIN	Function
-----	-----
10	→ speed[] size n = 10
8	→ speed = [8, 3, 6, 3, 2, 2, 4, 8, 1, 6]
3	
6	
3	
2	
2	
4	
8	
1	
6	
7	→ pos = 7

Sample Output 1

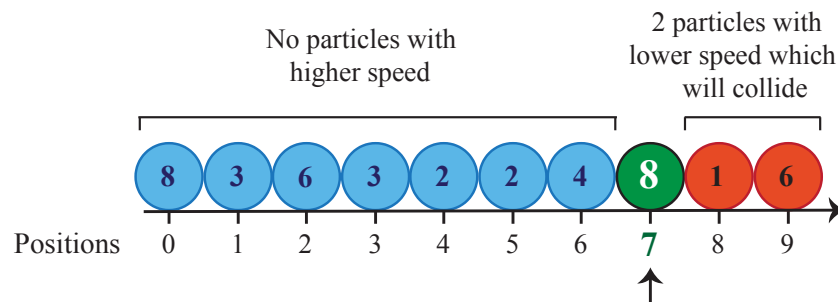
2

Explanation 1

$n = 10$

$speed = [8, 3, 6, 3, 2, 2, 4, 8, 1, 6]$

$pos = 7$



The particle at position $pos = 7$ will only collide with particles 8 and 9 as it passes them from behind. There are no other particles moving fast enough to collide with particle 7.

▼ Sample Case 2

Sample Input 2

STDIN	Function
-----	-----
6	→ speed[] size n = 6
1	→ speed = [1, 3, 7, 4, 6, 4]
3	
7	
4	
6	
4	
3	→ pos = 3

3

Sample Output 2

1

Explanation 2

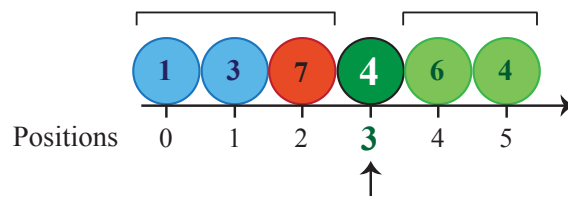
$n = 6$

$speed = [1, 3, 7, 4, 6, 4]$

$pos = 3$

1 particle with
higher speed
which will collide

No particles
with
lower speed



Particles 0 and 1 are moving slower, so they will never catch up. Particle 3 is moving at less than or equal to the speeds of particles ahead of it, numbers 4 and 5, so it can never catch up. The only collision with particle 3 occurs for particle 2.

INTERVIEWER GUIDELINES

▼ Hint 1

Calculate the number of particles that collide with the particle at the position 'pos'. Similarly, calculate the number of particles that the particle at position 'pos' collides with.

▼ Hint 2

A particle 'x' collides with particle 'y' if the speed of particle 'x' is strictly higher than the speed of particle 'y'.

▼ Solution

Concepts Covered: Basic Programming Skills, Loops, Arrays, Problem Solving. The problem tests the candidate's ability to use loops and array handling using comparison operators. It requires the candidate to come up with an algorithm to find the number of particles that collide with a given particle in constrained time complexity.

Optimal Solution:

Count the number of particles to the left of the particle at the position 'pos' whose speed is strictly higher than speed[pos]. Count the number of particles to the right whose speed is strictly smaller than speed[pos]. The sum of both these counts is the answer.

```
def collision(speed, pos):
    # Write your code here
    sp = speed[pos]
    ans = 0
    # to the left of speed[pos], count higher speeds
    for s in speed[:pos]:
        if s > sp: ans += 1
    # to the right of speed[pos], count lower speeds
    for s in speed[pos+1:]:
        if s < sp: ans += 1
    return ans
```

Error Handling:

1. It is important to be careful with the comparison operators and the condition that follow them.

▼ Complexity Analysis

Time Complexity - $O(n)$.

Since we iterate for all particles to the left and right of the particle at position 'pos'.

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

We can keep just a counter variable to maintain the counts of the particles that overtake the particle at pos or that are overtaken by the particle at pos.

▼ Follow up Question

Let's suppose you are provided with another constraint on the terminal point (X) for all the particles where they stop and the initial position of the particles. Now find the number of collisions that take place with the particle at position 'pos'.

Solution: This can be solved by checking for all other particles if they reach the terminal point earlier or later than the particle at 'pos'.

Pseudo Code -


```
def collision(speed, initial_pos, pos, X):
    # Write your code here
    cnt = 0
    for i in range (len(speed)):
        if((X - initial_pos[i]) / speed[i] < (X - initial_pos[pos]) /
speed[pos]):
            cnt++
    return cnt
```

CANDIDATE ANSWER

The candidate did not manually submit any code. The last compiled version has been auto-submitted and the score you see below is for the auto-submitted version.

Language used: **Java 8**

```
1  class Result {
2
3      /*
4       * Complete the 'collision' function below.
5       *
6       * The function is expected to return an INTEGER.
7       * The function accepts following parameters:
8       * 1. INTEGER_ARRAY speed
9       * 2. INTEGER pos
10      */
11
12      public static int collision(List<Integer> speed, int pos) {
13          int result = 0;
14          int hasilKurang = 0;
15
16          for(int i = 0; i < speed.size(); i++){
17              if(i + 1 < speed.size()){
18                  if(speed.get(i) > speed.get(i+1)){
19                      int tampHasilKurang = 0;
20                      tampHasilKurang = speed.get(i) - speed.get(i + 1);
21                      if(hasilKurang < tampHasilKurang){
22                          hasilKurang = tampHasilKurang;
23                          result = i;
24                      }
25                  }
26              }
27          }
28
29          return result;
30      }
31  }
32
33
34
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Test Case 0	Easy	Sample case	 Wrong Answer	0	0.1321 sec	30 KB

TestCase 0	Easy	Sample case	Wrong Answer	0	0.1321 sec	30 KB
TestCase 1	Easy	Sample case	Wrong Answer	0	0.1358 sec	29.9 KB
TestCase 2	Easy	Sample case	Wrong Answer	0	0.1199 sec	30 KB
TestCase 3	Medium	Sample case	Wrong Answer	0	0.1231 sec	29.9 KB
TestCase 4	Medium	Hidden case	Wrong Answer	0	0.1402 sec	30.1 KB
TestCase 5	Medium	Hidden case	Wrong Answer	0	0.1566 sec	32.4 KB
TestCase 6	Medium	Sample case	Wrong Answer	0	0.14 sec	30.8 KB
TestCase 7	Medium	Hidden case	Wrong Answer	0	0.2369 sec	43.3 KB
TestCase 8	Hard	Hidden case	Wrong Answer	0	0.3314 sec	44.3 KB
TestCase 9	Hard	Hidden case	Wrong Answer	0	0.3352 sec	54.8 KB
TestCase 10	Hard	Hidden case	Wrong Answer	0	0.3428 sec	57 KB
TestCase 11	Hard	Hidden case	Wrong Answer	0	0.3268 sec	56.2 KB
TestCase 12	Hard	Hidden case	Success	5	0.299 sec	56.7 KB
TestCase 13	Hard	Hidden case	Success	5	0.3743 sec	54.2 KB

No Comments

QUESTION 2



Correct Answer

Score 25

Growth in 2 Dimensions > Coding

Easy

Loops

Problem Solving

Arrays

Interviewer Guidelines

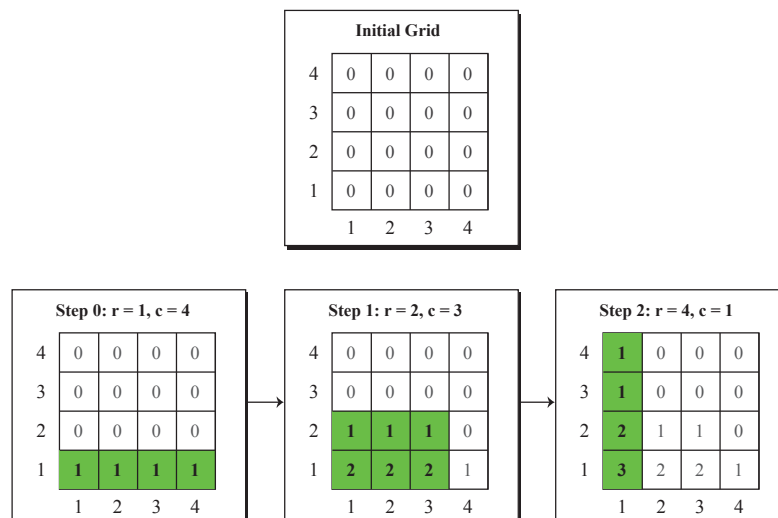
QUESTION DESCRIPTION

Start with an infinite two dimensional grid filled with zeros, indexed from $(1, 1)$ at the bottom left corner with coordinates increasing toward the top and right. Given a series of coordinates (r, c) , where r is the ending row and c is the ending column, add 1 to each element in the range from $(1, 1)$ to (r, c) inclusive. Once all coordinates are processed, determine how many cells contain the maximal value in the grid.

Example

$upRight = ["1\ 4", "2\ 3", "4\ 1"]$

The two space-separated integers within each string represent r and c respectively. The following diagrams show each iteration starting at zero. The maximal value in the grid is 3, and there is 1 occurrence at cell $(1, 1)$.



Function Description

Complete the function `countMax` in the editor below.

`countMax` has the following parameter(s):

string `upRight[n]`: an array of strings made of two space-separated integers, *r* and *c*.

Return

long: the number of occurrences of the final grid's maximal element

Constraints

- $1 \leq n \leq 100$
- $1 \leq \text{number of rows, number of columns} \leq 10^6$

▼ 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 `upRight`.
Each of the next *n* lines contains a string of two space-separated integers representing coordinates *r* and *c* for element `upRight[i]`.

▼ Sample Case 0

Sample Input

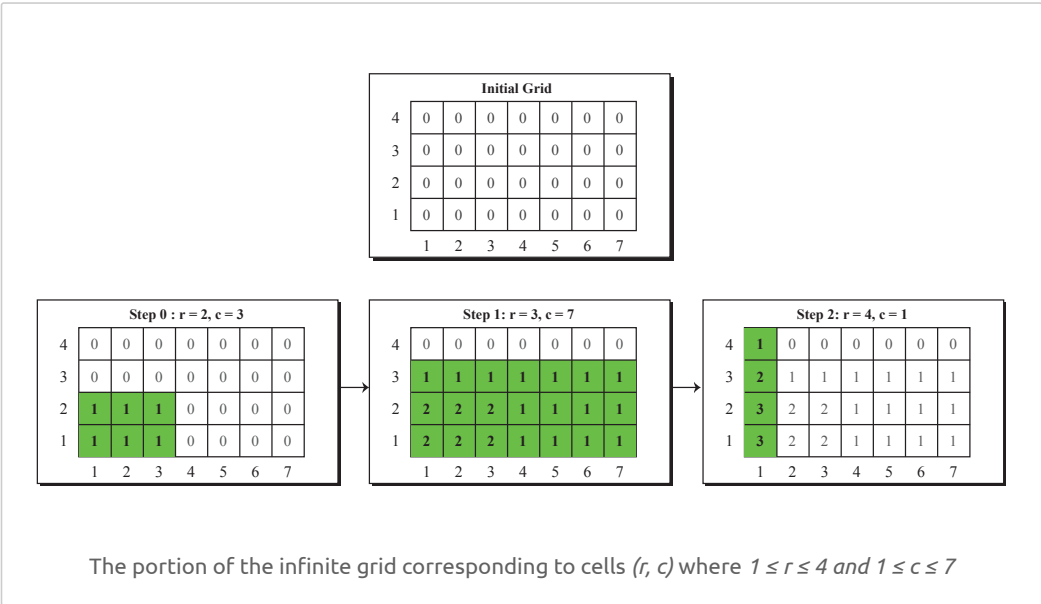
STDIN	Function
-----	-----
3 →	<code>upRight[]</code> size <code>n = 3</code>
2 3 →	<code>upRight = ['2 3', '3 7', '4 1']</code>
3 7	
4 1	

Sample Output

2

Explanation

Given `upRight = ["2 3", "3 7", "4 1"]`:



After processing all $n = 3$ coordinate pairs, the maximum value in any cell is 3. Because there are two

such cells with this maximal value, return 2 as the answer.

INTERVIEWER GUIDELINES

▼ Hint 1

Since number of rows and columns is of the order of 10^6 , the construction of the matrix is impossible. Calculate the answer using rows and columns separately.

▼ Hint 2

Use difference array to update the range $[1, \text{row}]$ and $[1, \text{column}]$ in each operation.

▼ Solution

Concepts Covered: Basic Programming Skills, Loops, Arrays, Prefix sums, Counting, Problem Solving. The problem tests the candidate's ability to use loops, array handling, and the difference arrays. It requires the candidate to come up with an algorithm to find the number of cells with the maximum value after a series of range submatrix updates in a constrained time and space complexity.

Optimal Solution: We don't need to construct the whole matrix since it would not fit into the required space complexity.

Since for a particular operation, the whole submatrix from $(1, 1)$ to (r, c) is updated by 1, we can calculate the number of times each row and column is updated. This can be done using difference array, which allows $O(1)$ update over a range $[l, r]$.

So let `rows[]` and `columns[]` be the two arrays of size = `mx` (where `mx = 10^6` the maximum number of rows and columns possible), which denote that how many times a particular row or column, represented by `rows[i]` and `columns[i]` is updated.

Let (r, c) be the current operation. We can do the following:

`rows[1]++`, `rows[r + 1]--`

`col[1]++`, `col[c + 1]--`

Finally we must take a prefix sum from left to right for each `rows[]` and `columns[]`.

So to find the number of cells with the maximum number(`max`), we can calculate `cnt_row` and `cnt_col`.

`cnt_row` = The number of cells `i`, in `rows[i]` such that `rows[i] = max`

`cnt_col` = The number of cells `i`, in `columns[i]` such that `columns[i] = max`

So the answer = `cnt_row * cnt_col`.

Time Complexity: $O(\max(\text{row}, \text{column}))$.

```
def countMax(upRight):
    n = len(upRight)
    # initialize arrays with zeros
    # size is fixed by constraints
    row = [0] * 1000005
    col = [0] * 1000005
    for i in range(n):
        # get the indices per query
        li = upRight[i].split(" ")
        # update appropriate rows and columns
        # for ranges where operations occur
        row[1] += 1
        row[int(li[0]) + 1] -= 1
        col[1] += 1
        col[int(li[1]) + 1] -= 1
    # calculate prefix sums by row and by column
    # while discovering the global maximum value
    sum1 = 0
    sum2 = 0
    mx = 0
    for i in range(1000005):
        sum1 += row[i]
        row[i] = sum1
        sum2 += col[i]
```

```

        col[i] = sum2
        mx = max(mx, row[i])
        mx = max(mx, col[i])
    # count the number of cells matching the global maximum
    cnt1 = 0
    cnt2 = 0
    for i in range(1000005):
        if(row[i] == mx):
            cnt1 += 1
        if(col[i] == mx):
            cnt2 += 1

    return cnt1 * cnt2

```

Brute Force Approach: Construct a matrix with the maximum possible rows and columns. For each query, update the whole submatrix from (1, 1) to (r, c) in $O(n^2)$ complexity. After all the updates, find the number of cells in the matrix with the maximum value by iterating through the whole matrix.

Time Complexity: $O(\text{rows} * \text{columns} * k)$, where the matrix has dimensions rows * columns, and k is the total number of operations.

Error Handling:

1. While updating the rows and columns, for implementing difference array trick, to update the range [l, r], its necessary to increment the index (r + 1) by 1 and not index r.
2. To count the value at each row and column index a prefix sum of both the difference arrays must be taken.
3. Since the maximum number of rows and columns is 10^6 , so candidates must be careful to declare the row and column count array as the maximum size only and not less than that.

▼ Complexity Analysis

Time Complexity - $O(mx)$, where $mx = 10^6$, the number of rows and columns possible.

Space Complexity - $O(mx)$.

▼ Follow up Question

Let's suppose we need now to count the number of cells in the matrix which have the minimum non-zero value.

Solution: We now count cnt_row and cnt_col, which denotes the number of rows and columns with value = min.

Pseudo Code -

```

def countMax(upRight):
    # Write your code here
    n = len(upRight)
    row = [0] * 1000005
    col = [0] * 1000005
    for i in range(n):
        li = upRight[i].split(" ")
        row[1] += 1
        row[int(li[0]) + 1] -= 1
        col[1] += 1
        col[int(li[1]) + 1] -= 1
    sum1 = 0
    sum2 = 0
    mn = 100000000
    for i in range(1000005):
        sum1 += row[i]
        row[i] = sum1
        sum2 += col[i]
        col[i] = sum2
        mn = min(mn, row[i])

```

```

        mn = min(mn, col[i])

    cnt1 = 0
    cnt2 = 0
    for i in range(1000005):
        if(row[i] == mn):
            cnt1 += 1
        if(col[i] == mn):
            cnt2 += 1

    return cnt1 * cnt2

```

▼ Follow up Question

Let's suppose now you are given queries Q, where you need to count the number of cells having value = Q[i].

Solution: After calculating rows[] and columns[], maintain two frequency arrays that denote the number of rows and columns which were updated freq[i] times.

So the solution to each query is ans[Q[i]] = freq_row[Q[i]] * freq_col[Q[i]].

Psuedo Code -

```

def countMax(upRight, query):
    # Write your code here
    n = len(upRight)
    row = [0] * 1000005
    col = [0] * 1000005
    for i in range(n):
        li = upRight[i].split(" ")
        row[1] += 1
        row[int(li[0]) + 1] -= 1
        col[1] += 1
        col[int(li[1]) + 1] -= 1
    sum1 = 0
    sum2 = 0
    mn = 100000000
    freq_row = [0] * (n + 1)
    freq_col = [0] * (n + 1)
    for i in range(1000005):
        sum1 += row[i]
        row[i] = sum1
        sum2 += col[i]
        col[i] = sum2
        freq_row[row[i]] += 1
        freq_col[col[i]] += 1

    ans = [0] * len(query)
    for i in range(len(query)):
        ans[i] = freq_row[query[i]] * freq_col[query[i]]

    return ans

```

CANDIDATE ANSWER

Language used: Java 8

```

1 class Result {
2
3     /*
4     * Complete the 'countMax' function below

```

```

4      * Complete the countMax function below.
5      *
6      * The function is expected to return a LONG_INTEGER.
7      * The function accepts STRING_ARRAY upRight as parameter.
8      */
9
10     public static long countMax(List<String> upRight) {
11         long result = 0;
12         int maxRow = 0;
13         int maxColumn = 0;
14
15         //menentukan maksimal baris dan kolom
16         for(int i = 0; i < upRight.size(); i++){
17             String[] tampSplit = upRight.get(i).split(" ");
18             int tampRow = Integer.parseInt(tampSplit[0]);
19             int tampColumn = Integer.parseInt(tampSplit[1]);
20
21             if(maxRow < tampRow){
22                 maxRow = tampRow;
23             } if(maxColumn < tampColumn){
24                 maxColumn = tampColumn;
25             }
26         }
27
28         //membuat array 2 dimensi
29         int[][] arr2d = new int[maxRow][maxColumn];
30
31         //mengisi array 2 dimensi
32         for(int i = 0; i < upRight.size(); i++){
33             String[] tampSplit = upRight.get(i).split(" ");
34             int tampRow = Integer.parseInt(tampSplit[0]);
35             int tampColumn = Integer.parseInt(tampSplit[1]);
36
37             for(int r = 0; r < tampRow; r++){
38                 for(int c = 0; c < tampColumn; c++){
39                     arr2d[r][c] += 1;
40                 }
41             }
42         }
43
44         /*
45         mencari nilai terbesar dari array 2 dimensi dan menghitung jumlah isi
46         array yang terdapat max value
47         */
48         int maxValue = 0;
49         for(int r = 0; r < maxRow; r++){
50             for(int c = 0; c < maxColumn; c++){
51                 if(maxValue < arr2d[r][c]){
52                     result = 0;
53                     maxValue = arr2d[r][c];
54                 }
55
56                 if(maxValue == arr2d[r][c]){
57                     result++;
58                 }
59             }
60         }
61
62
63
64         return result;
65     }
66
67 }

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	1	0.1215 sec	29.9 KB
Testcase 1	Easy	Sample case	✓ Success	5	0.1294 sec	29.9 KB
Testcase 2	Easy	Sample case	✓ Success	5	0.1318 sec	30 KB
Testcase 3	Easy	Sample case	✓ Success	5	0.165 sec	30.3 KB
Testcase 4	Easy	Sample case	✓ Success	5	1.3359 sec	145 KB
Testcase 5	Easy	Hidden case	✗ Runtime Error	0	1.2117 sec	500 KB
Testcase 6	Easy	Hidden case	✗ Runtime Error	0	0.7897 sec	463 KB
Testcase 7	Easy	Hidden case	✗ Runtime Error	0	0.9503 sec	460 KB
Testcase 8	Easy	Hidden case	✗ Runtime Error	0	0.9503 sec	464 KB
Testcase 9	Easy	Hidden case	✗ Runtime Error	0	0.8864 sec	482 KB
Testcase 10	Easy	Hidden case	✓ Success	4	0.1386 sec	30 KB

No Comments

QUESTION 3



Correct Answer

Score 25

Find the First Repeated Word in a Sentence > Coding Easy Strings Hashing

Problem Solving

QUESTION DESCRIPTION

A sentence is minimally defined as a word or group of words that ends with a period. A word is considered to be a sequence of letters delimited by a non-letter character. A *repeated word* is a case-sensitive word that appears more than once in a sentence (e.g.: 'had' = 'had' but 'had' ≠ 'Had'). Because substrings of a word are *not* delimited, they are *not* considered to be words, (e.g. 'hard' is not repeated in 'hardly'). Determine the first repeated word in a sentence.

Example

sentence = 'We work hard because hard work pays.'

The first matching word is 'hard'. Even though *work* is repeated and occurs first in the sentence, the first repetition is the word 'hard'.

Delimiters include the following:

- Whitespace: tab, space
- Others: , : ; - .

Function Description

Complete the function *firstRepeatedWord* in the editor below.

firstRepeatedWord has the following parameter(s):

string *sentence*: a sentence to analyze

Returns:

string: the first repeated word in sentence *s*

Constraints

- $0 < \text{length of sentence} < 1024$

- The following characters are delimiters between words: space, tab, comma (,), colon (:), semicolon (;), dash (-), and period (.).
- It is guaranteed that *sentence* contains one or more repeated words.
- Each word is separated by one or more delimiters.

▼ Input Format for Custom Testing

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

The only line contains a string, *sentence*.

▼ Sample Case

Sample Input

STDIN	Function
-----	-----
He had had quite enough of this nonsense.	→ sentence = 'He had had quite enough of this nonsense.'

Sample Output

had

Explanation

In this case, 'had' is the first and only word to appear twice in the sentence.

CANDIDATE ANSWER

Language used: **Java 8**

```

1  class Result {
2
3      /*
4       * Complete the 'firstRepeatedWord' function below.
5       *
6       * The function is expected to return a STRING.
7       * The function accepts STRING sentence as parameter.
8       */
9
10     public static String firstRepeatedWord(String sentence) {
11         String result = "";
12         String[] split = sentence.replace("\\s", " ")
13                                 .split(" ");
14         int length = split.length;
15
16         for(int i = 0; i < length; i++){
17             for(int j = 1; j < length; j++){
18                 if(split[i].equals(split[j]) && !split[i].equalsIgnoreCase("
19 ") && !split[j].equalsIgnoreCase(" ")) {
20                     result = split[i];
21                     break;
22                 }
23             }
24             if(!result.equalsIgnoreCase("")) {
25                 break;
26             }
27         }
28     }

```

```

28
29         return result;
30     }
31
32 }
33

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Medium	Sample case	✓ Success	1	0.0817 sec	23.8 KB
Testcase 1	Medium	Sample case	✓ Success	14	0.071 sec	23.6 KB
Testcase 2	Easy	Hidden case	✓ Success	10	0.0675 sec	23.5 KB
Testcase 3	Easy	Hidden case	✗ Wrong Answer	0	0.0692 sec	23.8 KB
Testcase 4	Medium	Hidden case	✗ Wrong Answer	0	0.0702 sec	23.7 KB

No Comments

QUESTION 4



Not Submitted

Score 0

How many Sentences ?

> Coding

Strings

Medium

Data Structures

Algorithms

Problem Solving

Hash Map

Combinatorics

QUESTION DESCRIPTION

Given an array of words and an array of sentences, determine which words are anagrams of each other. Calculate how many sentences can be created by replacing any word with one of its anagrams.

Example

wordSet = ['listen', 'silent', 'it', 'is']

sentence = 'listen it is silent'

Determine that *listen* is an anagram of *silent*. Those two words can be replaced with their anagrams. The four sentences that can be created are:

- listen it is silent
- listen it is listen
- silent it is silent
- silent it is listen

Function Description

Complete the *countSentences* function in the editor below.

countSentences has the following parameters:

string wordSet[n]: an array of strings

string sentences[m]: an array of strings

Returns:

int[]: an array of *m* integers that denote the number of sentences that can be formed from each sentence

Constraints

- $0 < n \leq 10^5$
- $1 \leq \text{length of each word} \leq 20$
- $1 \leq m \leq 1000$
- $3 \leq \text{words in a sentence} \leq 20$

▼ Input Format For Custom Testing

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

The first line contains an integer n denoting the number of elements in `wordSet[]`.

Each of the next n lines contains a string `wordSet[i]`.

The next line contains an integer m denoting the number of elements in `sentences[]`.

Each of the next m lines contains a string `sentences[i]` made up of a number of words separated by spaces.

▼ Sample Case 0

Sample Input

```
STDIN      Function
-----
6          → wordSet[] size n = 6
the        → wordSet = ['the', 'bats', 'tabs', 'in', 'cat', 'act']
bats
tabs
in
cat
act
3          → sentences[] size m = 3
cat the bats → sentences = ['cat the bats', 'in the act', 'act tabs in']
in the act
act tabs in
```

Sample Output

```
4
2
4
```

Explanation

Sentence 1: For the sentence '*cat the bats*', the sentences that can be formed are:

- cat the bats
- act the bats
- cat the tabs
- act the tabs

Sentence 2: For the sentence '*in the act*', the sentences that can be formed are:

- in the act
- in the cat

Sentence 3: For the sentence '*act tabs in*', the sentences that can be formed are:

- act tabs in
- cat tabs in
- act bats in
- cat bats in

Therefore, the integer array returned is [4, 2, 4].

INTERVIEWER GUIDELINES

▼ Hint 1

How can you easily check if two strings are anagrams? Answer: You can sort their characters. After sorting they must be identical.

▼ Hint 2

What is an efficient data structure to count the number of anagrams of a string? You can store the

frequency of each sorted string in a hash table.

▼ Solution

Concepts covered: Hash Table, Strings

Optimal Solution:

There are different methods to find whether two strings are anagrams or not. The easiest one is to just sort characters in both of them lexicographically. After sorting, these strings must be identical for them to be anagrams. Store each string in a hash map and count their frequencies. Now for each sentence, we need to iterate over every word and find the number of strings in the wordSet which are anagrams of this word. For each word in the sentence, sort it and then find its frequency in the hash table, call it $\text{freq}[i]$. The number of different sentences possible is the product of all the $\text{freq}[i]$'s: $\text{freq}[1] * \text{freq}[2] * \dots * \text{freq}[k]$ where k is the number of words in the sentence.

Implementation:

```
def countSentences(wordSet, sentences):
    freq = {}
    for word in wordSet:
        word = list(word)
        word.sort()
        word = ''.join(word)
        freq[word] = freq.get(word, 0) + 1
    ans = []
    for sentence in sentences:
        count = 1
        sentence = sentence.split()
        for word in sentence:
            word = list(word)
            word.sort()
            word = ''.join(word)
            count = count * freq[word]
        ans.append(count)
    return ans
```

Error Handling: Possible overflow in solutions using 32-bit integers for storing the of number of sentences.

▼ Complexity Analysis

Time Complexity - $O(n|w|^2 \log|w| + mk|w|)$, where $|w|$ is the average word length and k is the average number of words in a sentence

Sorting a string of length $|w|$: $O(|w| \log|w|)$

Sorting all the strings in wordSet: $O(n|w| \log|w|)$

Storing a string of length $|w|$ in the hash table: $O(|w|)$

Therefore, overall preprocess takes $O(n|w|^2 \log|w|)$ time

Getting the count of string of length $|w|$ from a hash table: $O(|w|)$

Getting the count of all the strings in a sentence from a hash table: $O(k|w|)$

Getting the answer for all the sentences: $O(mk|w|)$

Therefore, overall time complexity is $(n|w|^2 \log|w| + mk|w|)$

Space Complexity - $O(n|w|)$.

In the worst case, you may have to store all the strings in wordSet into the hash table, which takes $O(n|w|)$ space.

CANDIDATE ANSWER



No answer was submitted for this question. Showing compiled/saved versions.

Language used: Java 8

```

1  class Result {
2
3      /*
4       * Complete the 'countSentences' function below.
5       *
6       * The function is expected to return a LONG_INTEGER_ARRAY.
7       * The function accepts following parameters:
8       *   1. STRING_ARRAY wordSet
9       *   2. STRING_ARRAY sentences
10      */
11
12      public static List<Long> countSentences(List<String> wordSet,
13 List<String> sentences) {
14          // Write your code here
15
16      }
17
18  }
19

```

No Comments

QUESTION 5



Correct Answer

Score 25

Segment > Coding

Data Structures

Medium

Algorithms

Problem Solving

Stacks

Core Skills

QUESTION DESCRIPTION

We define a subarray of size x in an n -element array to be the contiguous block of elements in the inclusive range from index i to index j , where $j - i + 1 = x$ and $0 \leq i \leq j < n$.

For example, given array $arr = [8, 2, 4]$, the subarrays of size $x = 2$ would be $[8, 2]$ and $[2, 4]$. The minimum values of the two subarrays are $[2, 2]$. The maximum of those two minimum values is 2. This is the value you want to determine.

Function Description

Complete the function `segment` in the editor below. Your function must find the minimum value for each subarray of size x in array `arr` and return an integer denoting the *maximum* of these minima.

`segment` has the following parameter(s):

`x`: an integer, the segment length

`arr[arr[0],...arr[n-1]]`: an array of integers

Constraints

- $1 \leq n \leq 10^6$
- $1 \leq arr[i] \leq 10^9$
- $1 \leq x \leq 10^5$

▼ Input Format for Custom Testing

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

The first line contains an integer x .

The second line contains an integer n , the size of the array `arr`.

Each of the next n lines contains an integer `arr[i]`.

▼ Sample Case 0

Sample Input 0

```
1
5
1
2
3
1
2
```

Sample Output 1

```
3
```

Explanation 1

The following arguments are passed to your function:

$x = 1$

$arr = [1, 2, 3, 1, 2]$

The subarrays of size $x = 1$ are $[1]$, $[2]$, $[3]$, $[1]$, and $[2]$. Because each subarray only contains 1 element, each value is minimal with respect to the subarray it's in. We return the maximum of these values, which is 3 .

▼ Sample Case 1

Sample Input 1

```
2
3
1
1
1
```

Sample Output 1

```
1
```

Explanation 1

The following arguments are passed to your function:

$x = 2$

$arr = [1, 1, 1]$

The subarrays of size $x = 2$ are $[1, 1]$ and $[1, 1]$. The minimum value for both subarrays is 1 . We return the maximum of two 1 's, which is 1 .

▼ Sample Case 2

Sample Input 2

```
3
5
2
5
4
6
8
```

Sample Output 2

```
4
```

Explanation 2

The following arguments are passed to your function:

`x = 3`

`arr = [2, 5, 4, 6, 8]`

The subarrays of size `x = 3` are `[2, 5, 4]`, `[5, 4, 6]`, and `[4, 6, 8]`. The respective minimum values for the three subarrays are 2, 4, and 4. We return the maximum of these values, which is 4.

CANDIDATE ANSWER

Language used: Java 8

```
1  class Result {
2
3      /*
4       * Complete the 'segment' function below.
5       *
6       * The function is expected to return an INTEGER.
7       * The function accepts following parameters:
8       * 1. INTEGER x
9       * 2. INTEGER_ARRAY arr
10      */
11
12      public static int segment(int x, List<Integer> arr) {
13          //arr = [2, 5, 4, 6, 8]
14          int result = 0;
15          List<Integer[]> subArrays = new ArrayList<>();
16          Integer[] tampArr = new Integer[x];
17          int indexArrMax = 0;
18
19          if(x == 1){
20              for(int i = 0; i < arr.size(); i++){
21                  if(result < arr.get(i)){
22                      result = arr.get(i);
23                  }
24              }
25          } else{
26              for(int j = 0; j < arr.size(); j++){
27                  int index = 0;
28                  if(j + x <= arr.size()){
29                      for(int i = j; i < x; i++){
30                          tampArr[index] = arr.get(i);
31                          index++;
32                      }
33                      subArrays.add(tampArr);
34                  } else{
35                      break;
36                  }
37              }
38
39              //Menampung nilai minimum dari tiap subArray
40              Integer[] tampArrMax = new Integer[subArrays.size()];
41              for (Integer[] integer : subArrays) {
42                  int tampMinValue = 0;
43                  for(int i = 0; i < integer.length; i++){
44                      if(tampMinValue == 0){
45                          tampMinValue = integer[i];
46                      }
47
48                      if(tampMinValue > integer[i]){
```

```

49         tampMinValue = integer[i];
50     }
51 }
52 //System.out.println("indexArrMax: " + indexArrMax + " || " +
53 tampMinValue);
54     tampArrMax[indexArrMax] = tampMinValue;
55     indexArrMax++;
56 }
57
58 //Mencari nilai maksimum dari tampArrMax
59 for(int i = 0; i < tampArrMax.length; i++){
60     if(result < tampArrMax[i]){
61         result = tampArrMax[i];
62     }
63 }
64 }
65
66
67     return result;
68 }
69
70 }
71

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	✔ Success	1	0.1268 sec	30.3 KB
TestCase 1	Easy	Sample case	✔ Success	1	0.1391 sec	30.2 KB
TestCase 2	Easy	Sample case	✔ Success	1	0.1228 sec	29.9 KB
TestCase 3	Easy	Hidden case	✘ Wrong Answer	0	0.1294 sec	30 KB
TestCase 4	Easy	Hidden case	✔ Success	3	0.1438 sec	30.3 KB
TestCase 5	Easy	Hidden case	✔ Success	3	0.1458 sec	30.4 KB
TestCase 6	Medium	Hidden case	✘ Wrong Answer	0	0.1338 sec	30.4 KB
TestCase 7	Medium	Hidden case	✘ Wrong Answer	0	0.15 sec	30.5 KB
TestCase 8	Medium	Hidden case	✔ Success	6	0.2363 sec	36.1 KB
TestCase 9	Hard	Hidden case	✘ Wrong Answer	0	0.1824 sec	34.4 KB
TestCase 10	Hard	Hidden case	✘ Wrong Answer	0	0.1914 sec	36.1 KB
TestCase 11	Hard	Hidden case	✔ Success	10	0.8634 sec	44.5 KB
TestCase 12	Hard	Hidden case	✘ Terminated due to timeout	0	4.0113 sec	187 KB

No Comments