Full Name:	NGUYEN DANG VINH
Email:	dangvinhprovn@gmail.com
Test Name:	**Mock Test**
Taken On:	23 Feb 2024 20:25:46 IST
Time Taken:	0 min 42 sec/ 90 min
Invited by:	Ankush
Invited on:	23 Feb 2024 20:25:35 IST
Skills Score:	
Tags Score:	Algorithms 290/290 Arrays 95/95 Core CS 290/290 Data Structures 215/215 Easy 95/95 Medium 75/75 Queues 120/120 Search 75/75 Sorting 95/95 Strings 95/95 problem-solving 170/170

100%

290/290

scored in **Mock Test** in 0 min 42 sec on 23 Feb 2024 20:25:46 IST

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	Truck Tour > **Coding**	12 sec	120/ 120	✓
Q2	Pairs > **Coding**	11 sec	75/ 75	✓
Q3	Big Sorting > **Coding**	11 sec	95/ 95	✓

QUESTION 1

Correct Answer

Score 120

Truck Tour > Coding

Algorithms | Data Structures | Queues | Core CS

QUESTION DESCRIPTION

Suppose there is a circle. There are N petrol pumps on that circle. Petrol pumps are numbered 0 to $(N-1)$ (both inclusive). You have two pieces of information corresponding to each of the petrol pump: (1)

the amount of petrol that particular petrol pump will give, and (2) the distance from that petrol pump to the next petrol pump.

Initially, you have a tank of infinite capacity carrying no petrol. You can start the tour at any of the petrol pumps. Calculate the first point from where the truck will be able to complete the circle. Consider that the truck will stop at each of the petrol pumps. The truck will move one kilometer for each litre of the petrol.

Input Format

The first line will contain the value of N.
The next N lines will contain a pair of integers each, i.e. the amount of petrol that petrol pump will give and the distance between that petrol pump and the next petrol pump.

Constraints:

$1 \leq N \leq 10^5$
$1 \leq \text{amount of petrol, distance} \leq 10^9$

Output Format

An integer which will be the smallest index of the petrol pump from which we can start the tour.

Sample Input

```
3
1 5
10 3
3 4
```

Sample Output

```
1
```

Explanation

We can start the tour from the second petrol pump.

CANDIDATE ANSWER

Language used: **JavaScript (Node.js)**

```
1
2  /*
3   * Complete the 'truckTour' function below.
4   *
5   * The function is expected to return an INTEGER.
6   * The function accepts 2D_INTEGER_ARRAY petrolpumps as parameter.
7   */
8
9  /**
10  *
11  * @param {number[][]} petrolpumps
12  * @returns {number}
13  */
14 function truckTour(petrolpumps) {
15   let minIdxPetrolPumps = undefined;
16
17   let currentPetrol = 0;
18   //   let needAmountOfPetrol = 0;
19
20   for (let idx = 0; idx < petrolpumps.length; idx++) {
21     const [amountOfPetrol, distance] = petrolpumps[idx];
22     currentPetrol += amountOfPetrol - distance;
23
```

```
    if (currentPetrol < 0) {
      //   needAmountOfPetrol += currentPetrol;
      minIdxPetrolPumps = undefined;
      currentPetrol = 0;
    } else if (currentPetrol > 0 && minIdxPetrolPumps === undefined) {
      minIdxPetrolPumps = idx;
    }
  }

  //   currentPetrol + needAmountOfPetrol need to bigger 0.
  //   console.log({ currentPetrol, needAmountOfPetrol, minIdxPetrolPumps });

  return minIdxPetrolPumps;
}

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	Success	0	0.0375 sec	41.6 KB
Testcase 2	Easy	Hidden case	Success	10	0.0418 sec	42.4 KB
Testcase 3	Easy	Hidden case	Success	10	0.0484 sec	42.6 KB
Testcase 4	Easy	Hidden case	Success	10	0.0579 sec	42.6 KB
Testcase 5	Easy	Hidden case	Success	10	0.1556 sec	74.5 KB
Testcase 6	Easy	Hidden case	Success	10	0.153 sec	73.8 KB
Testcase 7	Easy	Hidden case	Success	10	0.1351 sec	74.3 KB
Testcase 8	Easy	Hidden case	Success	10	0.1446 sec	74.4 KB
Testcase 9	Easy	Hidden case	Success	10	0.1658 sec	73.2 KB
Testcase 10	Easy	Hidden case	Success	10	0.1857 sec	74.2 KB
Testcase 11	Easy	Hidden case	Success	10	0.299 sec	73.5 KB
Testcase 12	Easy	Hidden case	Success	10	0.1478 sec	73.8 KB
Testcase 13	Easy	Hidden case	Success	10	0.1831 sec	73.8 KB

No Comments

QUESTION 2

Correct Answer

Score 75

Pairs › Coding

Search | Algorithms | Medium | problem-solving | Core CS

QUESTION DESCRIPTION

Given an array of integers and a target value, determine the number of pairs of array elements that have a difference equal to the target value.

Example

$k = 1$

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

There are three values that differ by $k = 1$: $2 - 1 = 1$, $3 - 2 = 1$, and $4 - 3 = 1$. Return 3.

Function Description

Complete the *pairs* function below.

pairs has the following parameter(s):

- *int k:* an integer, the target difference
- *int arr[n]:* an array of integers

Returns

- *int:* the number of pairs that satisfy the criterion

Input Format

The first line contains two space-separated integers n and k, the size of arr and the target value.
The second line contains n space-separated integers of the array arr.

Constraints

- $2 \leq n \leq 10^5$
- $0 < k < 10^9$
- $0 < arr[i] < 2^{31} - 1$
- each integer $arr[i]$ will be unique

Sample Input

```
STDIN       Function
-----       --------
5 2         arr[] size n = 5, k =2
1 5 3 4 2   arr = [1, 5, 3, 4, 2]
```

Sample Output

```
3
```

Explanation

There are 3 pairs of integers in the set with a difference of 2: [5,3], [4,2] and [3,1]. .

CANDIDATE ANSWER

Language used: **JavaScript (Node.js)**

```
1
2  /*
3   * Complete the 'pairs' function below.
4   *
5   * The function is expected to return an INTEGER.
6   * The function accepts following parameters:
7   *  1. INTEGER k
8   *  2. INTEGER_ARRAY arr
9   */
10
11 /**
12  *
13  * @param {number} k
14  * @param {number[]} arr
15  * @returns {number}
16  */
17 function pairs(k, arr) {
18   /** @type {Object<number, number>} */
19   const mapNumer = {};
20   let numberOfPairs = 0;
21
22   for (const num of arr) {
23     mapNumer[num] ? mapNumer[num]++ : (mapNumer[num] = 1);
24   }
25
26   while (arr.length) {
27     const num = arr.pop();
28     const difference = num - k;
29     console.log(mapNumer[difference]);
```

```
    if (mapNumer[difference]) numberOfPairs += mapNumer[difference];
  }

  return numberOfPairs;
}

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Hidden case	Success	5	0.046 sec	42.5 KB
Testcase 2	Easy	Hidden case	Success	5	0.0535 sec	43.1 KB
Testcase 3	Easy	Hidden case	Success	5	1.1885 sec	42.2 KB
Testcase 4	Easy	Hidden case	Success	5	0.0474 sec	42.6 KB
Testcase 5	Easy	Hidden case	Success	5	0.0931 sec	43.1 KB
Testcase 6	Easy	Hidden case	Success	5	0.1281 sec	51.2 KB
Testcase 7	Easy	Hidden case	Success	5	0.1923 sec	51.5 KB
Testcase 8	Easy	Hidden case	Success	5	1.2996 sec	48.9 KB
Testcase 9	Easy	Hidden case	Success	5	0.0914 sec	50.8 KB
Testcase 10	Easy	Hidden case	Success	5	0.2217 sec	51.9 KB
Testcase 11	Easy	Hidden case	Success	5	0.9397 sec	79.2 KB
Testcase 12	Easy	Hidden case	Success	5	1.9491 sec	79.1 KB
Testcase 13	Easy	Hidden case	Success	5	0.8703 sec	80.7 KB
Testcase 14	Easy	Hidden case	Success	5	2.2146 sec	79.5 KB
Testcase 15	Easy	Hidden case	Success	5	0.8457 sec	80.9 KB
Testcase 16	Easy	Sample case	Success	0	0.0452 sec	42.1 KB
Testcase 17	Easy	Sample case	Success	0	0.0645 sec	41.6 KB
Testcase 18	Easy	Sample case	Success	0	0.0706 sec	41.4 KB

No Comments

QUESTION 3

Correct Answer

Score 95

Big Sorting › Coding

Sorting | Strings | Algorithms | Easy | Data Structures | Arrays | problem-solving | Core CS

QUESTION DESCRIPTION

Consider an array of numeric strings where each string is a positive number with anywhere from 1 to 10^6 digits. Sort the array's elements in *non-decreasing*, or ascending order of their integer values and return the sorted array.

Example

$unsorted = [\text{'1'}, \text{'200'}, \text{'150'}, \text{'3'}]$

Return the array ['1', '3', '150', '200'].

Function Description

Complete the *bigSorting* function in the editor below.

bigSorting has the following parameter(s):

- *string unsorted[n]:* an unsorted array of integers as strings

Returns

- *string[n]:* the array sorted in numerical order

Input Format

The first line contains an integer, n, the number of strings in $unsorted$.
Each of the n subsequent lines contains an integer string, $unsorted[i]$.

Constraints

- $1 \leq n \leq 2 \times 10^5$
- Each string is guaranteed to represent a positive integer.
- There will be no leading zeros.
- The total number of digits across all strings in $unsorted$ is between 1 and 10^6 (inclusive).

Sample Input 0

```
6
3141592653589793238462643383279 5
1
3
10
3
5
```

Sample Output 0

```
1
3
3
5
10
3141592653589793238462643383279 5
```

Explanation 0

The initial array of strings is
$unsorted = [3141592653589793238462643383279 5, 1, 3, 10, 3, 5]$. When we order each string by the real-world integer value it represents, we get:

$$1 \leq 3 \leq 3 \leq 5 \leq 10 \leq 3141592653589793238462643383279 5$$

We then print each value on a new line, from smallest to largest.

Sample Input 1

```
8
1
2
100
12303479849857341718340192371
3084193741082937
3084193741082938
111
200
```

Sample Output 1

```
1
2
100
111
200
3084193741082937
```

```
3084193741082938
123034798498573417183401923 71
```

CANDIDATE ANSWER

Language used: **JavaScript (Node.js)**

```
1
2  /*
3   * Complete the 'bigSorting' function below.
4   *
5   * The function is expected to return a STRING_ARRAY.
6   * The function accepts STRING_ARRAY unsorted as parameter.
7   */
8
9  /**
10  *
11  * @param {string[]} unsorted
12  * @returns {bigint[]}
13  */
14 function bigSorting(unsorted) {
15   const sorted = unsorted
16     .map((string) => BigInt(string))
17     .sort((a, b) => (a > b ? 1 : a < b ? -1 : 0));
18
19   return sorted;
20 }
21
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	Success	0	0.0422 sec	41.9 KB
Testcase 2	Medium	Hidden case	Success	10	0.0435 sec	41.6 KB
Testcase 3	Medium	Hidden case	Success	10	0.0577 sec	45.6 KB
Testcase 4	Hard	Hidden case	Success	15	0.5898 sec	56.8 KB
Testcase 5	Hard	Hidden case	Success	15	1.7763 sec	57.8 KB
Testcase 6	Hard	Hidden case	Success	15	0.5266 sec	56.2 KB
Testcase 7	Hard	Hidden case	Success	15	0.6258 sec	59.1 KB
Testcase 8	Hard	Hidden case	Success	15	0.4967 sec	93.1 KB
Testcase 9	Easy	Sample case	Success	0	1.207 sec	41.1 KB

No Comments