Choose a Flask

An olive oil seller needs to measure oil for customers using only one type of flask. There are many flasks available, each with markings at various levels. Each customer must receive a flask filled to a mark that is at least equal to the amount ordered. Given a list of customer requirements and a list of flasks with their measurements, determine the single type of flask that will result in minimal loss to the merchant. Loss is the sum of *marking - requirement* for each order. Return the zero-based index of the flask type chosen. If there are multiple answers, return the minimum index. If no flask will satisfy the constraints, return *-1*.

For example, there are $n = 4$ orders for *requirements = [4, 6, 6, 7]* units of oil. There are $m = 3$ types of flasks available with *markings = ["3 5 7", "6 8 9", "3 5 6"]*. These *markings* are given as 2D array with *total_marks* rows and 2 columns, the first is the index of the flask and second the mark. To fill the orders using the flask at *markings[0] = [3, 5, 7]*, the loss is calculated as *marking - requirement* for each order so, *5 - 4 = 1, 7 - 6 = 1, 7 - 6 = 1* and *7 - 7 = 0*. The total loss then is *1 + 1 + 1 + 0 = 3*. Choosing the flask at *markings[1]*, the loss is *6 - 4 = 2, 6 - 6 = 0, 6 - 6 = 0, 8 - 7 = 1 -> 2 + 0 + 0 + 1 = 3*. The third flask has a maximum mark at *6* which is smaller than the largest order, so it cannot be used. In this case, flask type *0* is chosen.

NOTE: The markings 2D array will be given in order of the flasks, i.e., the markings for the 0-index flask will be followed by markings of 1-index flask and so on. For each flask, the given markings will also be in the sorted order.

Function Description

Complete the function *chooseFlask* in the editor below. The function must return an integer, the index of the flask to choose or *-1* if none will work.

chooseFlask has the following parameter(s):

requirements[requirements[0],...requirements[n-1]]: an array of integers that denote the requirements of the customers.

m: an integer, the number of flasks.

markings[markings[0],...markings[total_marks-1]]: a 2D array of integers with 2 columns, the first column signifies the index of the flask and second signifies one mark.

Constraints

- $1 \le n \le 10^5$
- $1 \le m \le 10^4$
- $1 \le total\_marks \le 10^5$
- $1 \le requirements[i] \le 10^9$ *(where $0 \le i < n$)*
- $0 \le markings[i][0] < m$ *(where $0 \le i < total\_marks$)*
- $0 \le markings[i][1] \le 10^9$ *(where $0 \le i < total\_marks$)*

Input Format For Custom Testing

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

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

The next line contains an integer, *m*, the number of flasks.

The next line contains an integer, *total_marks*, the number of rows in *markings*

The next line contains an integer, *2*, the number of columns in *markings*

Each line j of the *total_marks* subsequent lines (where $0 \le j < total\_marks$) contains two space separated integers that describe *markings[j]*.

Sample Case 0

Sample Input For Custom Testing

```
2
4
6
2
5
2
0 5
0 7
0 10
1 4
1 10
```

Sample Output

Explanation

~~The loss using the flask at index 0 is (5-4 + 7-6) = 2 and the loss using the flask at index 1 is (4-4 + 10-6) = 4. It is better to use flask 0.~~

Sample Case 1

Sample Input For Custom Testing

```
2
10
15
3
6
2
0 11
0 20
1 11
1 17
2 12
2 16
```

Sample Output

```
1
```

Explanation

The loss using the flasks are *6*, *3*, and *3* respectively. The flask with minimum loss and minimum index is *1*.

YOUR ANSWER

View Code Diff | Python 3

```
#!/bin/python3 ...

#
# Complete the 'chooseFlask' function below.
#
# The function is expected to return an INTEGER.
# The function accepts following parameters:
#  1. INTEGER_ARRAY requirements
#  2. INTEGER m
#  3. 2D_INTEGER_ARRAY markings
#

def chooseFlask(requirements, m, markings):
    # Write your code here

if __name__ == '__main__': ...
```

Line: 10 Col: 1

Test against custom input

Run Code | Submit code & Continue

(You can submit any number of times)

Download sample test cases *The input/output files have Unix line endings. Do not use Notepad to edit them on windows.*