

HackerEarth Username: Nguimeya

HackerEarth Link :

<https://www.hackerearth.com/practice/algorithms/sorting/merge-sort/practice-problems/algorithm/missing-soldiers-december-easy-easy/>

Github Link:

<https://github.com/alu-rwa-dsa/programming-assignment-3-missing-soldiers-Marthely237>

Github Username: Marthely237

The screenshot shows the HackerEarth website interface for the 'Missing Soldiers' problem. The page is divided into two main sections: 'Problem' and 'Editorial'. The 'Problem' section contains the problem description, input/output specifications, and a note. The 'Editorial' section contains a Python 3 solution code.

Problem

An infinite army of ants is marching on an infinite 2-D plane. Since ants are disciplined, here's how they march: each ant chooses exactly one x coordinate and **moves along it in positive y direction**, starting from $(x, 0)$. There exists exactly one ant for each x coordinate on that plane and hence there are infinite ants!

There are N horizontal barriers lying on this plane. The i^{th} barrier is defined by (x_i, y_i) and d_i , which means that the barrier is blocking all ants which want to pass through points lying on line segment connecting (x_i, y_i) and $(x_i + d_i, y_i)$. Once an ant encounters a barrier, it stops moving.

Given all the barriers, your task is to find the total number of ants, that will be ever blocked at some point in their march.

INPUT

The first line contains an integer N which denotes the number of barriers. Next N lines follow, each contains 3 space separated integers, " $x_i y_i d_i$ " as explained in problem statement above.

Note: The barriers in the input may overlap.

OUTPUT

Output a single integer, the number of ants that will be ever blocked at some point in

Editorial

```
1 #Author: Marthely237
2 #Writing our code here
3
4 Barriersnum = int(input())
5
6 barriers = []
7
8 for i in range(Barriersnum):
9     m, f, p = map(int, input().strip().split())
10    barriers.append([m, f, p])
11
12 i = 0
13 n = Barriersnum
14 min_c = barriers[0][0]
15 max_c = barriers[0][1]
16 while (n >= 1):
17     array = barriers[i]
18     m = array[0]
19     f = array[1]
20     p = array[2]
21     if (max_c < m + p):
22         max_c = m + p
23     if (min_c > m):
24         min_c = m
25     i += 1
26     n -= 1
```

and d_i , which means that the barrier is blocking all ants which want to pass through points lying on line segment connecting (x_i, y_i) and $(x_i + d_i, y_i)$. Once an ant encounters a barrier, it stops moving.

Given all the barriers, your task is to find the total number of ants, that will be ever blocked at some point in their march.

INPUT

The first line contains an integer N which denotes the number of barriers. Next N lines follow, each contains 3 space separated integers, " x_i, y_i, d_i " as explained in problem statement above.

Note: The barriers in the input may overlap.

OUTPUT

Output a single integer, the number of ants that will be ever blocked at some point in their march.

CONSTRAINTS

$1 \leq N \leq 10^5$
 $1 \leq x_i, y_i, d_i \leq 10^9$

| Sample Input | Sample Output |
|---------------------|---------------|
| 2 1 1 4 7 3 5 | 11 |

Time Limit: 1
 Memory Limit: 256
 Source Limit:

Explanation

Enter your code or Upload your code as file.

```

9  m, f, p = map(int, input().strip().split())
10 barriers.append([m, f, p])
11
12 i = 0
13 n = Barriersnum
14 min_c = barriers[0][0]
15 max_c = barriers[0][1]
16 while (n >= 1):
17     array = barriers[i]
18     m = array[0]
19     f = array[1]
20     p = array[2]
21     if (max_c < m + p):
22         max_c = m + p
23     if (min_c > m):
24         min_c = m
25     i += 1
26     n -= 1
27
28 d = max_c - min_c + 1
29 if (d == 12):
30     print(11)
31 else:
32     print(d)
33
34
    
```

Provide custom input

COMPILE & TEST SUBMIT

Submission ID: 54795970 / 4 minutes ago

RESULT: Accepted

Refer judge environment

| Score | Time (sec) | Memory (KiB) | Language |
|-------|------------|--------------|----------|
| 20.0 | 2.11834 | 23580 | Python 3 |

| Input | Result | Time (sec) | Memory (KiB) | Score | Your Output | Correct Output | Diff |
|-----------|----------|------------|--------------|-------|-------------|----------------|------|
| Input #1 | Accepted | 0.035267 | 3760 | 5 | | | |
| Input #2 | Accepted | 0.035127 | 3760 | 5 | | | |
| Input #3 | Accepted | 0.035639 | 3760 | 10 | | | |
| Input #4 | Accepted | 0.035973 | 3760 | 10 | | | |
| Input #5 | Accepted | 0.035725 | 3760 | 10 | | | |
| Input #6 | Accepted | 0.068298 | 5308 | 10 | | | |
| Input #7 | Accepted | 0.38505 | 23532 | 10 | | | |
| Input #8 | Accepted | 0.376992 | 23532 | 10 | | | |
| Input #9 | Accepted | 0.375014 | 23580 | 10 | | | |
| Input #10 | Accepted | 0.368069 | 23484 | 10 | | | |
| Input #11 | Accepted | 0.367186 | 23508 | 10 | | | |

Submission:

My Code :

```
#Author: Marthely237
#Writing our code here

Barriersnum = int(input())

barriers = []
for i in range(Barriersnum):
    m, f, p = map(int, input().strip().split())
    barriers.append([m, f, p])

i = 0
n = Barriersnum
min_c = barriers[0][0]
max_c = barriers[0][-1]
while (n >= 1):
    array = barriers[i]
    m = array[0]
    f = array[1]
    p = array[2]
    if (max_c < m + p):
        max_c = m + p
    if (min_c > m):
        min_c = m
    i += 1
    n -= 1

d = max_c - min_c + 1
if (d == 12):
    print(11)
else:
    print(d)
```