Problem 2276: Count Integers in Intervals

Problem Information

Difficulty: Hard
Acceptance Rate: 35.28%
Paid Only: No
Tags: Design, Segment Tree, Ordered Set

Problem Description

Given an **empty** set of intervals, implement a data structure that can:

* **Add** an interval to the set of intervals. * **Count** the number of integers that are present in **at least one** interval.

Implement the `CountIntervals` class:

* `CountIntervals()` Initializes the object with an empty set of intervals. * `void add(int left, int right)` Adds the interval `[left, right]` to the set of intervals. * `int count()` Returns the number of integers that are present in **at least one** interval.

Note that an interval `[left, right]` denotes all the integers `x` where `left <= x <= right`.

Example 1:

Input ["CountIntervals", "add", "add", "count", "add", "count"] [[], [2, 3], [7, 10], [], [5, 8], []] **Output** [null, null, null, 6, null, 8] **Explanation** CountIntervals countIntervals = new CountIntervals(); // initialize the object with an empty set of intervals. countIntervals.add(2, 3); // add [2, 3] to the set of intervals. countIntervals.add(7, 10); // add [7, 10] to the set of intervals. countIntervals.count(); // return 6 // the integers 2 and 3 are present in the interval [2, 3]. // the integers 7, 8, 9, and 10 are present in the interval [7, 10]. countIntervals.add(5, 8); // add [5, 8] to the set of intervals. countIntervals.count(); // return 8 // the integers 2 and 3 are present in the interval [2, 3]. // the integers 5 and 6 are present in the interval [5, 8]. // the integers 7 and 8 are present in the intervals [5, 8] and [7, 10]. // the integers 9 and 10 are present in the interval [7, 10].

Constraints:

* `1 <= left <= right <= 109` * At most `105` calls **in total** will be made to `add` and `count`.
* At least **one** call will be made to `count`.

Code Snippets

C++:

```
class CountIntervals {
public:
CountIntervals() {

}

void add(int left, int right) {

}

int count() {

}
};

/**
* Your CountIntervals object will be instantiated and called as such:
* CountIntervals* obj = new CountIntervals();
* obj->add(left,right);
* int param_2 = obj->count();
*/
```

Java:

```
class CountIntervals {

public CountIntervals() {

}

public void add(int left, int right) {
```

```
}

public int count() {

}
}

/**
* Your CountIntervals object will be instantiated and called as such:
* CountIntervals obj = new CountIntervals();
* obj.add(left,right);
* int param_2 = obj.count();
*/
```

Python3:

```
class CountIntervals:

def __init__(self):


def add(self, left: int, right: int) -> None:


def count(self) -> int:



# Your CountIntervals object will be instantiated and called as such:
# obj = CountIntervals()
# obj.add(left,right)
# param_2 = obj.count()
```