Problem 715: Range Module

Problem Information

Difficulty: Hard
Acceptance Rate: 44.40%
Paid Only: No
Tags: Design, Segment Tree, Ordered Set

Problem Description

A Range Module is a module that tracks ranges of numbers. Design a data structure to track the ranges represented as **half-open intervals** and query about them.

A **half-open interval** `[left, right)` denotes all the real numbers `x` where `left <= x < right`.

Implement the `RangeModule` class:

* `RangeModule()` Initializes the object of the data structure. * `void addRange(int left, int right)` Adds the **half-open interval** `[left, right)`, tracking every real number in that interval. Adding an interval that partially overlaps with currently tracked numbers should add any numbers in the interval `[left, right)` that are not already tracked. * `boolean queryRange(int left, int right)` Returns `true` if every real number in the interval `[left, right)` is currently being tracked, and `false` otherwise. * `void removeRange(int left, int right)` Stops tracking every real number currently being tracked in the **half-open interval** `[left, right)`.

Example 1:

Input ["RangeModule", "addRange", "removeRange", "queryRange", "queryRange", "queryRange"] [[], [10, 20], [14, 16], [10, 14], [13, 15], [16, 17]] **Output** [null, null, null, true, false, true] **Explanation** RangeModule rangeModule = new RangeModule(); rangeModule.addRange(10, 20); rangeModule.removeRange(14, 16); rangeModule.queryRange(10, 14); // return True,(Every number in [10, 14) is being tracked) rangeModule.queryRange(13, 15); // return False,(Numbers like 14, 14.03, 14.17 in [13, 15) are not being tracked) rangeModule.queryRange(16, 17); // return True, (The number 16 in [16, 17) is still being tracked, despite the remove operation)

Constraints:

* `1 <= left < right <= 109` * At most `104` calls will be made to `addRange`, `queryRange`, and `removeRange`.

Code Snippets

C++:

```
class RangeModule {
public:
RangeModule() {

}

void addRange(int left, int right) {

}

bool queryRange(int left, int right) {

}

void removeRange(int left, int right) {

}
};

/**
* Your RangeModule object will be instantiated and called as such:
* RangeModule* obj = new RangeModule();
* obj->addRange(left,right);
* bool param_2 = obj->queryRange(left,right);
* obj->removeRange(left,right);
*/
```

Java:

```
class RangeModule {

public RangeModule() {
}

public void addRange(int left, int right) {

}

public boolean queryRange(int left, int right) {

}

public void removeRange(int left, int right) {

}
}

/**
* Your RangeModule object will be instantiated and called as such:
* RangeModule obj = new RangeModule();
* obj.addRange(left,right);
* boolean param_2 = obj.queryRange(left,right);
* obj.removeRange(left,right);
*/
```

Python3:

```
class RangeModule:

def __init__(self):


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


def queryRange(self, left: int, right: int) -> bool:


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



# Your RangeModule object will be instantiated and called as such:
```

```
# obj = RangeModule()
# obj.addRange(left,right)
# param_2 = obj.queryRange(left,right)
# obj.removeRange(left,right)
```