Problem 352: Data Stream as Disjoint Intervals

Problem Information

Difficulty: Hard
Acceptance Rate: 59.74%
Paid Only: No
Tags: Hash Table, Binary Search, Union Find, Design, Data Stream, Ordered Set

Problem Description

Given a data stream input of non-negative integers `a1, a2, ..., an`, summarize the numbers seen so far as a list of disjoint intervals.

Implement the `SummaryRanges` class:

* `SummaryRanges()` Initializes the object with an empty stream. * `void addNum(int value)` Adds the integer `value` to the stream. * `int[][] getIntervals()` Returns a summary of the integers in the stream currently as a list of disjoint intervals `[starti, endi]`. The answer should be sorted by `starti`.

Example 1:

Input ["SummaryRanges", "addNum", "getIntervals", "addNum", "getIntervals", "addNum", "getIntervals", "addNum", "getIntervals", "addNum", "getIntervals"] [[], [1], [], [3], [], [7], [], [2], [], [6], []] **Output** [null, null, [[1, 1]], null, [[1, 1], [3, 3]], null, [[1, 1], [3, 3], [7, 7]], null, [[1, 3], [7, 7]], null, [[1, 3], [6, 7]]] **Explanation** SummaryRanges summaryRanges = new SummaryRanges(); summaryRanges.addNum(1); // arr = [1] summaryRanges.getIntervals(); // return [[1, 1]] summaryRanges.addNum(3); // arr = [1, 3] summaryRanges.getIntervals(); // return [[1, 1], [3, 3]] summaryRanges.addNum(7); // arr = [1, 3, 7] summaryRanges.getIntervals(); // return [[1, 1], [3, 3], [7, 7]] summaryRanges.addNum(2); // arr = [1, 2, 3, 7] summaryRanges.getIntervals(); // return [[1, 3], [7, 7]] summaryRanges.addNum(6); // arr = [1, 2, 3, 6, 7] summaryRanges.getIntervals(); // return [[1, 3], [6, 7]]

Constraints:

* `0 <= value <= 104` * At most `3 * 104` calls will be made to `addNum` and `getIntervals`. * At most `102` calls will be made to `getIntervals`.

Follow up: What if there are lots of merges and the number of disjoint intervals is small compared to the size of the data stream?

Code Snippets

C++:

```
class SummaryRanges {
public:
SummaryRanges() {

}

void addNum(int value) {

}

vector<vector<int>> getIntervals() {

}
};

/**
* Your SummaryRanges object will be instantiated and called as such:
* SummaryRanges* obj = new SummaryRanges();
* obj->addNum(value);
* vector<vector<int>> param_2 = obj->getIntervals();
*/
```

Java:

```
class SummaryRanges {

public SummaryRanges() {

}

public void addNum(int value) {
```

```
}

public int[][] getIntervals() {

}
}

/**
* Your SummaryRanges object will be instantiated and called as such:
* SummaryRanges obj = new SummaryRanges();
* obj.addNum(value);
* int[][] param_2 = obj.getIntervals();
*/
```

Python3:

```
class SummaryRanges:

def __init__(self):


def addNum(self, value: int) -> None:


def getIntervals(self) -> List[List[int]]:



# Your SummaryRanges object will be instantiated and called as such:
# obj = SummaryRanges()
# obj.addNum(value)
# param_2 = obj.getIntervals()
```