Problem 432: All O`one Data Structure

Problem Information

Difficulty: Hard
Acceptance Rate: 44.07%
Paid Only: No
Tags: Hash Table, Linked List, Design, Doubly-Linked List

Problem Description

Design a data structure to store the strings' count with the ability to return the strings with minimum and maximum counts.

Implement the `AllOne` class:

* `AllOne()` Initializes the object of the data structure. * `inc(String key)` Increments the count of the string `key` by `1`. If `key` does not exist in the data structure, insert it with count `1`. * `dec(String key)` Decrements the count of the string `key` by `1`. If the count of `key` is `0` after the decrement, remove it from the data structure. It is guaranteed that `key` exists in the data structure before the decrement. * `getMaxKey()` Returns one of the keys with the maximal count. If no element exists, return an empty string `""`. * `getMinKey()` Returns one of the keys with the minimum count. If no element exists, return an empty string `""`.

Note that each function must run in `O(1)` average time complexity.

Example 1:

Input ["AllOne", "inc", "inc", "getMaxKey", "getMinKey", "inc", "getMaxKey", "getMinKey"] [[], ["hello"], ["hello"], [], [], ["leet"], [], []] **Output** [null, null, null, "hello", "hello", null, "hello", "leet"] **Explanation** AllOne allOne = new AllOne(); allOne.inc("hello"); allOne.inc("hello"); allOne.getMaxKey(); // return "hello" allOne.getMinKey(); // return "hello" allOne.inc("leet"); allOne.getMaxKey(); // return "hello" allOne.getMinKey(); // return "leet"

Constraints:

* `1 <= key.length <= 10` * `key` consists of lowercase English letters. * It is guaranteed that for each call to `dec`, `key` is existing in the data structure. * At most `5 * 104` calls will be made to `inc`, `dec`, `getMaxKey`, and `getMinKey`.

Code Snippets

C++:

```
class AllOne {
public:
AllOne() {

}

void inc(string key) {

}

void dec(string key) {

}

string getMaxKey() {

}

string getMinKey() {

}
};

/**
* Your AllOne object will be instantiated and called as such:
* AllOne* obj = new AllOne();
* obj->inc(key);
* obj->dec(key);
* string param_3 = obj->getMaxKey();
* string param_4 = obj->getMinKey();
*/
```

Java:

```
class AllOne {

public AllOne() {

}

public void inc(String key) {

}

public void dec(String key) {

}

public String getMaxKey() {

}

public String getMinKey() {

}
}

/**
* Your AllOne object will be instantiated and called as such:
* AllOne obj = new AllOne();
* obj.inc(key);
* obj.dec(key);
* String param_3 = obj.getMaxKey();
* String param_4 = obj.getMinKey();
*/
```

Python3:

```
class AllOne:

def __init__(self):



def inc(self, key: str) -> None:



def dec(self, key: str) -> None:
```

```
def getMaxKey(self) -> str:


def getMinKey(self) -> str:



# Your AllOne object will be instantiated and called as such:
# obj = AllOne()
# obj.inc(key)
# obj.dec(key)
# param_3 = obj.getMaxKey()
# param_4 = obj.getMinKey()
```