432. All O`one Data Structure

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 $5 * 10^4$ calls will be made to `inc`, `dec`, `getMaxKey`, and `getMinKey`.