Data Structure Design...

Video - 17

Leetcode - 432

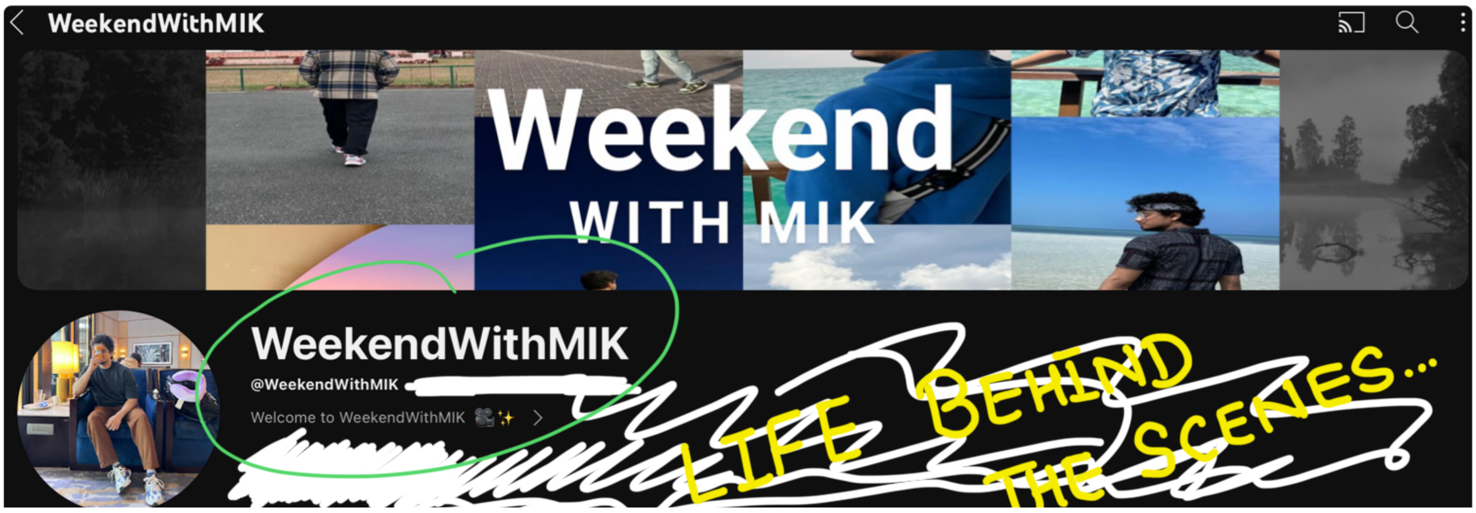
Hard

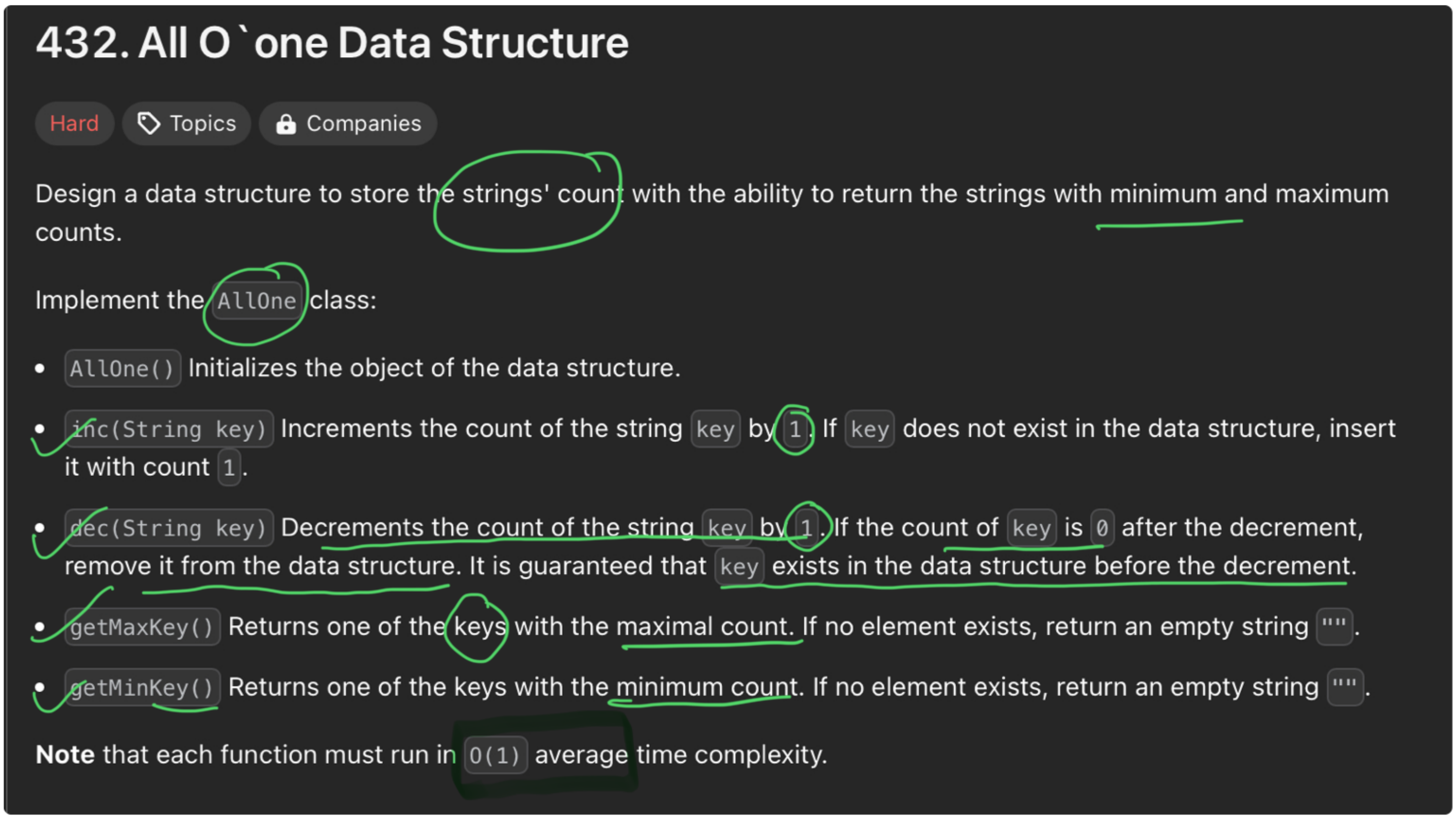
Facebook, Instagram → codestorywithMIK

(Twitter) → CSwithMIK

codestorywithMIK → WhatsApp

Something Big Coming on 100K :)





432. All O`one Data Structure

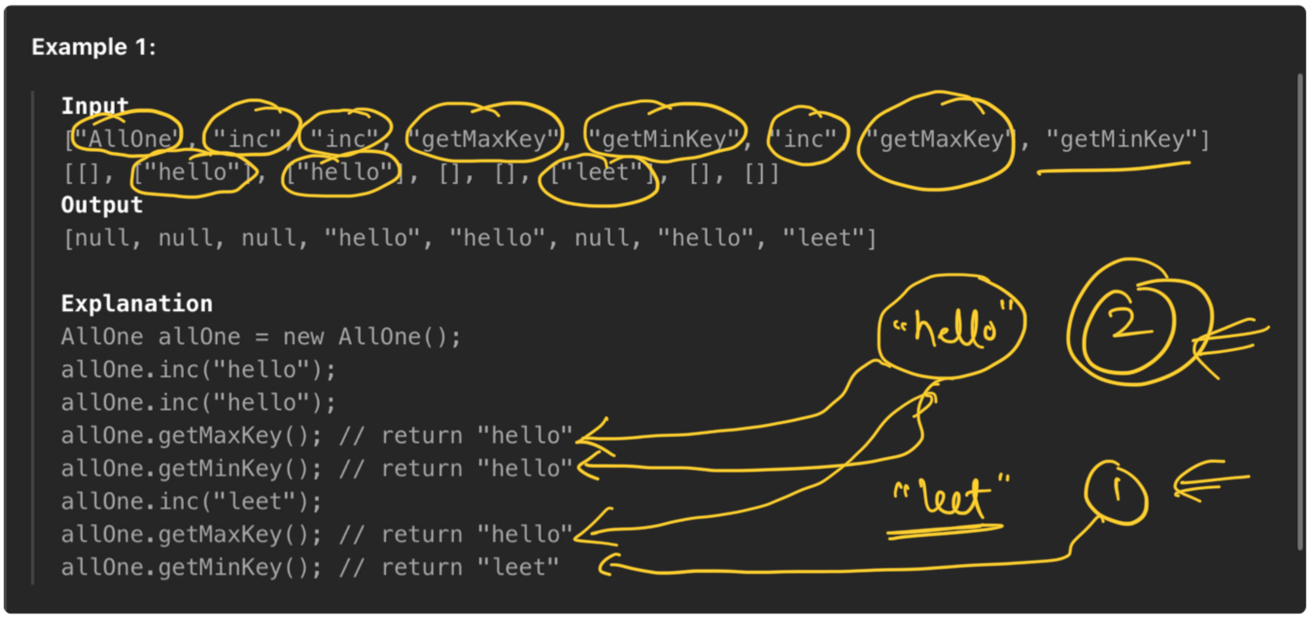
Hard · Topics · Companies

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.

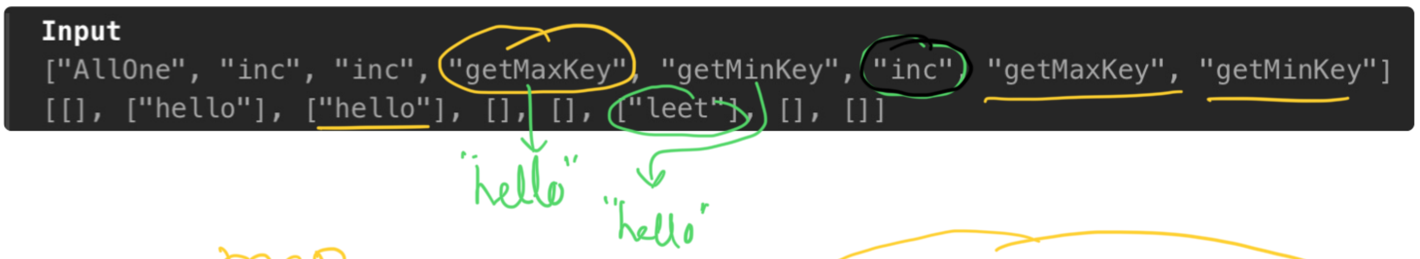


Example 1:

```
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"
```

Let's Build Thought Process....



```
Input
["AllOne", "inc", "inc", "getMaxKey", "getMinKey", "inc", "getMaxKey", "getMinKey"]
[[], ["hello"], ["hello"], [], [], ["leet"], [], []]
```

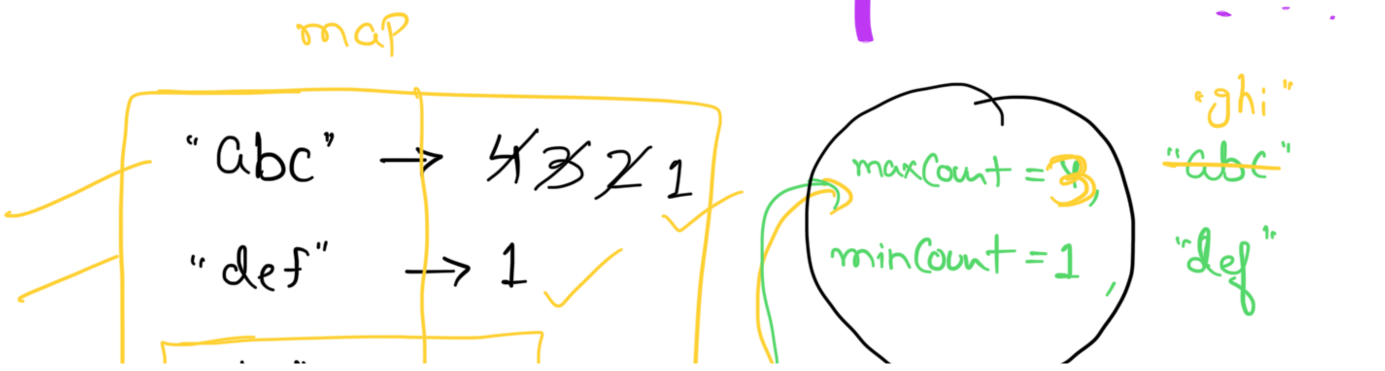
map

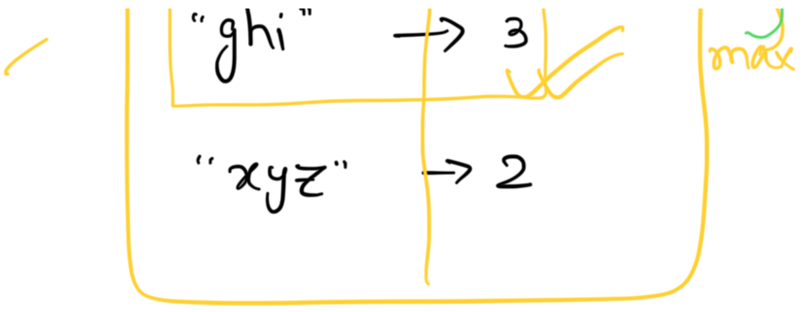
string	count
"hello"	2
"leet"	1

maxCount = 2 , "hello"

minCount = 1 , "leet"

What's the problem???





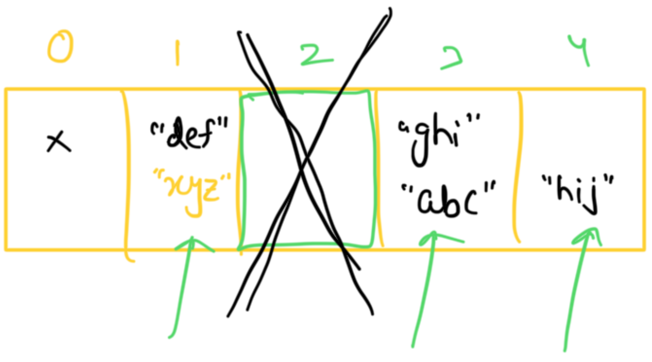
~~$O(n)$.~~

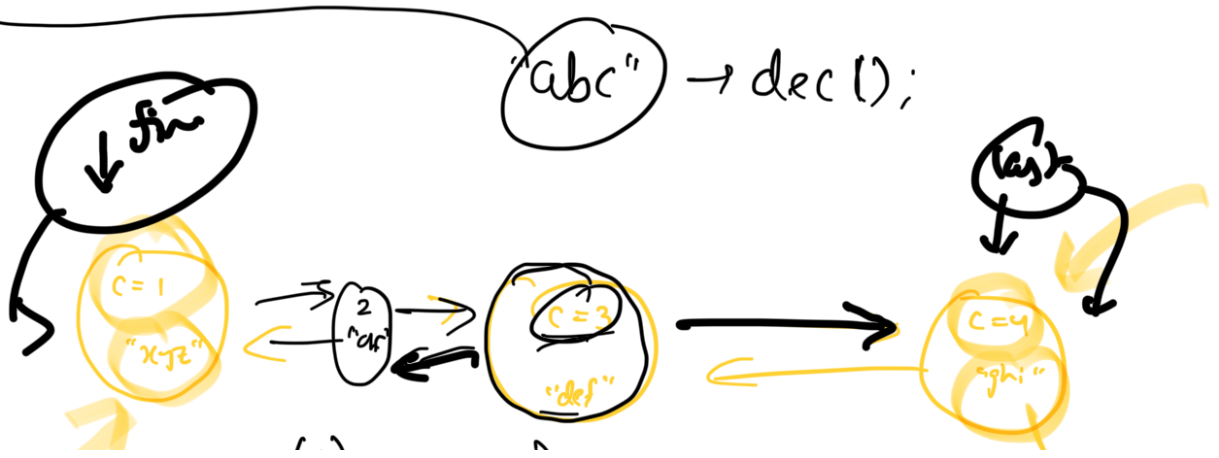
$O(1)$

map

"abc"	ADDR
"def"	1
"ghi"	3
"xyz"	1
"hij"	4

start idx = 1
end-idx = 4





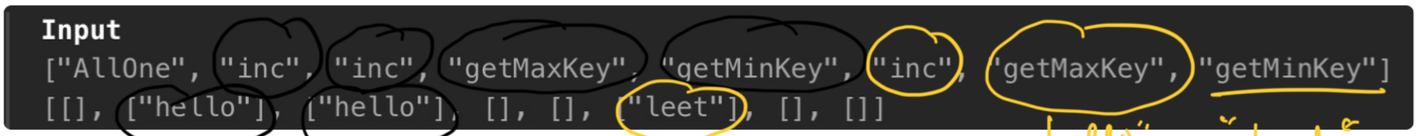
O(1)   ADDR

"abc"

Double Linked list -

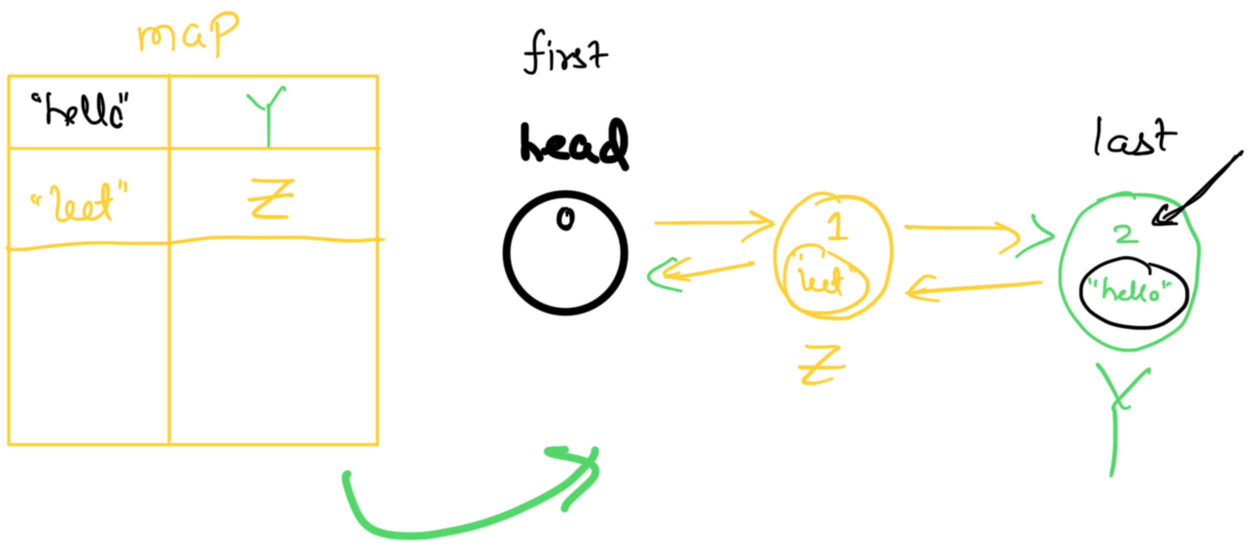
LRU
LFU

std:: list < >



```
Input
["AllOne", "inc", "inc", "getMaxKey", "getMinKey", "inc", "getMaxKey", "getMinKey"]
[[], ["hello"], ["hello"], [], [], ["leet"], [], []]
```

"hello"   "leet"



node {

```
Node {
        int count =
        list<string> keys;
        Node* next;
        Node* prev;
}
```