# Problem 895: Maximum Frequency Stack

## Problem Information

**Difficulty:** Hard
**Acceptance Rate:** 66.52%
**Paid Only:** No
**Tags:** Hash Table, Stack, Design, Ordered Set

## Problem Description

Design a stack-like data structure to push elements to the stack and pop the most frequent element from the stack.

Implement the `FreqStack` class:

* `FreqStack()` constructs an empty frequency stack. * `void push(int val)` pushes an integer `val` onto the top of the stack. * `int pop()` removes and returns the most frequent element in the stack. * If there is a tie for the most frequent element, the element closest to the stack's top is removed and returned.

**Example 1:**

**Input** ["FreqStack", "push", "push", "push", "push", "push", "push", "pop", "pop", "pop", "pop"] [[], [5], [7], [5], [7], [4], [5], [], [], [], []] **Output** [null, null, null, null, null, null, null, 5, 7, 5, 4] **Explanation** FreqStack freqStack = new FreqStack(); freqStack.push(5); // The stack is [5] freqStack.push(7); // The stack is [5,7] freqStack.push(5); // The stack is [5,7,5] freqStack.push(7); // The stack is [5,7,5,7] freqStack.push(4); // The stack is [5,7,5,7,4] freqStack.push(5); // The stack is [5,7,5,7,4,5] freqStack.pop(); // return 5, as 5 is the most frequent. The stack becomes [5,7,5,7,4]. freqStack.pop(); // return 7, as 5 and 7 is the most frequent, but 7 is closest to the top. The stack becomes [5,7,5,4]. freqStack.pop(); // return 5, as 5 is the most frequent. The stack becomes [5,7,4]. freqStack.pop(); // return 4, as 4, 5 and 7 is the most frequent, but 4 is closest to the top. The stack becomes [5,7].

**Constraints:**

* `0 <= val <= 109` * At most `2 * 104` calls will be made to `push` and `pop`. * It is guaranteed that there will be at least one element in the stack before calling `pop`.

## Code Snippets

**C++:**

```
class FreqStack {
public:
FreqStack() {

}

void push(int val) {

}

int pop() {

}
};

/**
* Your FreqStack object will be instantiated and called as such:
* FreqStack* obj = new FreqStack();
* obj->push(val);
* int param_2 = obj->pop();
*/
```

**Java:**

```
class FreqStack {

public FreqStack() {

}

public void push(int val) {

}
```

```
public int pop() {

}
}

/**
* Your FreqStack object will be instantiated and called as such:
* FreqStack obj = new FreqStack();
* obj.push(val);
* int param_2 = obj.pop();
*/
```

**Python3:**

```
class FreqStack:

def __init__(self):


def push(self, val: int) -> None:


def pop(self) -> int:



# Your FreqStack object will be instantiated and called as such:
# obj = FreqStack()
# obj.push(val)
# param_2 = obj.pop()
```