Problem 155: Min Stack

Problem Information

Difficulty: Medium
Acceptance Rate: 57.28%
Paid Only: No
Tags: Stack, Design

Problem Description

Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.

Implement the `MinStack` class:

* `MinStack()` initializes the stack object. * `void push(int val)` pushes the element `val` onto the stack. * `void pop()` removes the element on the top of the stack. * `int top()` gets the top element of the stack. * `int getMin()` retrieves the minimum element in the stack.

You must implement a solution with `O(1)` time complexity for each function.

Example 1:

Input ["MinStack","push","push","push","getMin","pop","top","getMin"] [[],[-2],[0],[-3],[],[],[],[]] **Output** [null,null,null,null,-3,null,0,-2] **Explanation** MinStack minStack = new MinStack(); minStack.push(-2); minStack.push(0); minStack.push(-3); minStack.getMin(); // return -3 minStack.pop(); minStack.top(); // return 0 minStack.getMin(); // return -2

Constraints:

* `-231 <= val <= 231 - 1` * Methods `pop`, `top` and `getMin` operations will always be called on **non-empty** stacks. * At most `3 * 104` calls will be made to `push`, `pop`, `top`, and `getMin`.

Code Snippets

C++:

```
class MinStack {
public:
MinStack() {

}

void push(int val) {

}

void pop() {

}

int top() {

}

int getMin() {

}
};

/**
* Your MinStack object will be instantiated and called as such:
* MinStack* obj = new MinStack();
* obj->push(val);
* obj->pop();
* int param_3 = obj->top();
* int param_4 = obj->getMin();
*/
```

Java:

```
class MinStack {

public MinStack() {

}
```

```
public void push(int val) {

}

public void pop() {

}

public int top() {

}

public int getMin() {

}
}

/**
* Your MinStack object will be instantiated and called as such:
* MinStack obj = new MinStack();
* obj.push(val);
* obj.pop();
* int param_3 = obj.top();
* int param_4 = obj.getMin();
*/
```

Python3:

```
class MinStack:

def __init__(self):


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


def pop(self) -> None:


def top(self) -> int:
```

```
def getMin(self) -> int:



# Your MinStack object will be instantiated and called as such:
# obj = MinStack()
# obj.push(val)
# obj.pop()
# param_3 = obj.top()
# param_4 = obj.getMin()
```