### 155 Min Stack

2018年4月3日 13:49

## **Question:**

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

- push(x) -- Push element x onto stack.
- pop() -- Removes the element on top of the stack.
- top() -- Get the top element.
- getMin() -- Retrieve the minimum element in the stack.

#### **Example:**

```
MinStack minStack = new MinStack();
minStack.push(-2);
minStack.push(0);
minStack.push(-3);
minStack.getMin(); --> Returns -3.
minStack.pop();
minStack.top(); --> Returns 0.
minStack.getMin(); --> Returns -2.
```

来自 < https://leetcode.com/problems/min-stack/description/>

设计一个支持 push, pop, top 操作,并能在常量时间内检索最小元素的栈。

- push(x) -- 将元素x推入栈中。
- pop() -- 删除栈顶的元素。
- top() -- 获取栈顶元素。
- getMin() -- 检索栈中的最小元素。

# **Solution for Python3:**

```
1
    from collections import deque
 2
     class MinStack1:
 3
         def __init__(self):
 4
 5
             initialize your data structure here.
 6
 7
             self.d1 = deque()
             self.d2 = deque()
 8
 9
10
         def push(self, x):
11
12
13
             :type x: int
             :rtype: void
14
15
16
             self.d1.append(x)
             if (not self.d2 or x <= self.getMin()):</pre>
17
                 self.d2.append(x)
18
19
```

```
20
         def pop(self):
21
22
              :rtype: void
23
24
             if self.d1[-1] == self.getMin():
25
                 self.d2.pop()
26
             self.d1.pop()
27
         def top(self):
28
29
30
              :rtype: int
31
32
             return self.d1[-1]
33
34
         def getMin(self):
35
36
37
              :rtype: int
              0.000
38
39
             return self.d2[-1]
40
41
42
     class MinStack2:
43
44
         def __init__(self):
45
46
             initialize your data structure here.
47
48
             self.L = []
49
50
51
         def push(self, x):
52
53
              :type x: int
54
              :rtype: void
55
56
             self.L.append((x, min(self.getMin(), x)))
57
58
59
         def pop(self):
60
61
              :rtype: void
62
63
             self.L.pop()
64
65
         def top(self):
66
67
68
              :rtype: int
```

```
.....
69
             if self.L:
70
71
                return self.L[-1][0]
72
             return None
73
74
75
        def getMin(self):
76
77
             :rtype: int
78
79
             if self.L:
80
                return self.L[-1][1]
81
             return sys.maxsize
82
83
84
   # Your MinStack object will be instantiated and called as such:
85
   # obj = MinStack()
   # obj.push(x)
86
87
   # obj.pop()
88  # param 3 = obj.top()
   # param 4 = obj.getMin()
89
```

## Solution for C++:

```
class MinStack1 {
 1
 2
    private:
 3
        stack<int> s1, s2;
 4
    public:
 5
         /** initialize your data structure here. */
 6
         MinStack() {
 7
 8
         }
 9
         void push(int x) {
10
11
             s1.push(x);
             if (s2.empty() || x <= getMin()) {</pre>
12
13
                s2.push(x);
14
             }
         }
15
16
         void pop() {
17
18
             if (s1.top() == getMin) {
19
                s2.pop();
20
             }
21
             s1.pop();
22
         }
23
24
         int top() {
```

```
25
             return s1.top();
26
        }
27
        int getMin() {
28
29
             return s2.top();
30
        }
31
    };
32
33
34
    class MinStack2 {
35
    private:
36
        vector<pair<int, int>> v;
37
    public:
        /** initialize your data structure here. */
38
39
        MinStack() {
40
41
        }
42
43
        void push(int x) {
44
            v.push back(make pair(x, min(x, getMin())));
45
        }
46
47
        void pop() {
48
            v.pop back();
49
        }
50
51
        int top() {
52
            if (!v.empty())
53
                return v.back().first;
54
                // return (*(v.end() - 1)).first;
55
             return NULL;
56
        }
57
58
        int getMin() {
59
            if (!v.empty())
60
                return v.back().second;
61
                // return (*(v.end() - 1)).second;
62
             return INT_MAX;
63
        }
64
    };
65
66
67
68
    * Your MinStack object will be instantiated and called as such:
69
     * MinStack obj = new MinStack();
     * obj.push(x);
70
71
     * obj.pop();
72
     * int param 3 = obj.top();
73
     * int param 4 = obj.getMin();
```