

DST_3D_Dec 2023 3D

Min Stack - LeetCode

ChatGPT

https://leetcode.com/problems/min-stack/

Problem List

Run

Submit

0

Premium

Description

Note

Editorial

Solutions

Submissions

155. Min Stack

Attempted

Medium Topics Companies Hint

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.

13.5K 80 ☆ ↗ ⌚

<> Code

C Auto

```
1 #include<stdio.h>
2 #include<stdlib.h>
3
4 typedef struct{
5     int val;
6     int min;
7 }Node;
8
9
10 typedef struct {
11     Node* stack;
12     int top, capacity;
13 } MinStack;
14
15 MinStack*

```

Saved to local Ln 1, Col 1

Testcase

Test Result

Accepted Runtime: 3 ms

Case 1



Input





Watchlist Ideas





Search

ENG IN


13:04 13-01-2024

  Problem List < >

 Run  Submit  



   0  Premium





</> Code

C ▾  Auto


```
1  #include<stdio.h>
2  #include<stdlib.h>
3
4  typedef struct{
5      int val;
6      int min;
7  }Node;
8
9
10 typedef struct {
11     Node* stack;
12     int top, capacity;
13 } MinStack;
14
15
16
17 MinStack* minStackCreate() {
18     MinStack* stack=(MinStack*)malloc(sizeof(MinStack));
19     stack->stack=(Node*)malloc(sizeof(Node)*100);
20     stack->top=-1;
21     stack->capacity=100;
22     return stack;
23 }
24
25
```

Saved to localLn 16, Col 1

  Problem List < >

 Run  Submit  

</> Code

C ▾  Auto

```
25 void minStackPush(MinStack* obj, int val) {
26     if (obj->top == -1) {
27         obj->stack[++(obj->top)].val = val;
28         obj->stack[obj->top].min = val;
29     } else {
30         obj->stack[++(obj->top)].val = val;
31         obj->stack[obj->top].min = (val < obj->stack[obj->top - 1].min) ? val : obj->stack[obj->top - 1].min;
32     }
33 }
34
35 void minStackPop(MinStack* obj) {
36     if (obj->top >= 0) {
37         obj->top--;
38     }
39 }
40
41 int minStackTop(MinStack* obj) {
42     return obj->stack[obj->top].val;
43 }
44
45 int minStackGetMin(MinStack* obj) {
46     return obj->stack[obj->top].min;
47 }
48
49
```

Saved to local

</> Code

C   Auto

```
33 }
34
35 void minStackPop(MinStack* obj) {
36     if (obj->top >= 0) {
37         obj->top--;
38     }
39 }
40
41 int minStackTop(MinStack* obj) {
42     return obj->stack[obj->top].val;
43 }
44
45 int minStackGetMin(MinStack* obj) {
46     return obj->stack[obj->top].min;
47 }
48
49
50 void minStackFree(MinStack* obj) {
51     free(obj->stack);
52     free(obj);
53 }
54
55
56 /**
```

☒ Testcase |  Test Result

Accepted Runtime: 0 ms

• Case 1

Input

["MinStack","push","push","push","getMin","pop","top","getMin"]

[[],[-2],[0],[-3],[],[],[],[[]]]

Output

[null,null,null,null,-3,null,0,-2]

Expected

[null,null,null,null,-3,null,0,-2]

 [Contribute a testcase](#)