LeetCode Question 1-Lab 01

155. Min Stack

Solved

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.

Constraints:

- $-2^{31} <= val <= 2^{31} - 1$
- Methods `pop`, `top` and `getMin` operations will always be called on **non-empty** stacks.
- At most $3 * 10^4$ calls will be made to `push`, `pop`, `top`, and `getMin`.

```
typedef struct {
    int a[30000];
    int top;
} MinStack;
int c=0;

MinStack* minStackCreate() {
    MinStack* n=(MinStack*)malloc(sizeof(MinStack));
    n->top=-1;
    return n;
}

void minStackPush(MinStack* obj, int val) {
    c++;
```

LeetCode Question 1-Lab 01

```
    obj->a[++obj->top]=val;
}

void minStackPop(MinStack* obj) {
    obj->top--;c--;
}

int minStackTop(MinStack* obj) {
    return obj->a[obj->top];
}

int minStackGetMin(MinStack* obj) {
    int min=obj->a[0];
    if(obj->top==-1)
    return -1;
    for(int i=0;i<c;i++)
    {
        if(obj->a[i]<min)
        min=obj->a[i];
    }
    return min;
}

void minStackFree(MinStack* obj) {
    free(obj);
    obj=NULL;c=0;
}

/**
 * Your MinStack struct will be instantiated and called as such:
 * MinStack* obj = minStackCreate();
 * minStackPush(obj, val);

 * minStackPop(obj);

 * int param_3 = minStackTop(obj);

 * int param_4 = minStackGetMin(obj);

 * minStackFree(obj);
*/
```

LeetCode Question 1-Lab 01

Accepted

Madhushree S Shetty submitted at May 06, 2024 18:30

Editorial Solution

Runtime

59 ms

Beats 12.48% of users with C

Memory

13.54 MB

Beats 41.19% of users with C