155. Min Stack

Easy

1517161FavoriteShare

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.

class Node{

public:

int val;

Node\* next;

Node\* min;

Node(int v){

val=v;

next=NULL;

min=NULL;

}

};

class MinStack {

public:

/\*\* initialize your data structure here. \*/

Node\* head=NULL;

Node\* min=NULL;

MinStack() {

}

void push(int x) {

Node\* curr=new Node(x);

curr->next=head;

head=curr;

if(min==NULL||x<min->val){

curr->min=min;

min=curr;

}

}

void pop() {

Node\* curr=head;

head=head->next;

if(min==curr){

min=curr->min;

}

delete(curr);

}

int top() {

return head->val;

}

int getMin() {

return min->val;

}

};

/\*\*

\* Your MinStack object will be instantiated and called as such:

\* MinStack obj = new MinStack();

\* obj.push(x);

\* obj.pop();

\* int param\_3 = obj.top();

\* int param\_4 = obj.getMin();

\*/

Success

[Details](https://leetcode.com/submissions/detail/211551113/)

Runtime: 40 ms, faster than 93.53% of C++ online submissions for Min Stack.

Memory Usage: 17.2 MB, less than 9.14% of C++ online submissions forMin Stack.