Interview Cake

You want to be able to access the *largest element* in a stack.

Use the built-in `Stack` class to **implement a *new* class `MaxStack` with a function `getMax()` that returns the largest element in the stack.** `getMax()` should not remove the item.

Your stacks will contain only integers.

Gotchas

What if we push several items in increasing numeric order (like 1, 2, 3, 4...), so that there is a *new max* after each `push()`? What if we then `pop()` each of these items off, so that there is a *new max* after each `pop()`? Your algorithm shouldn't pay a steep cost in these edge cases.

You should be able to get a runtime of $O(1)$ for `push()`, `pop()`, and `getMax()`.

Breakdown

Solution

This part requires full access (/upgrade)

You've already used up your free full access questions!

If you subscribe to our weekly newsletter (/free-weekly-coding-interview-problem-newsletter), you'll get another free full access question every week.

To see the full solution and breakdown for *all* of our questions, you'll have to upgrade to full access (/upgrade).

Upgrade now ➜ (/upgrade)

Complexity

$O(1)$ time for `push()`, `pop()`, and `getMax()`. $O(m)$ additional space, where m is the number of operations performed on the stack.

What We Learned

This part requires full access (/upgrade)

You've already used up your free full access questions!

If you subscribe to our weekly newsletter (/free-weekly-coding-interview-problem-newsletter), you'll get another free full access question every week.

To see the full solution and breakdown for *all* of our questions, you'll have to upgrade to full access (/upgrade).

Upgrade now ➜ (/upgrade)

Want more coding interview help?

Check out **interviewcake.com** for more advice, guides, and practice questions.