Solution Review: Implementing Two Stacks Using One List

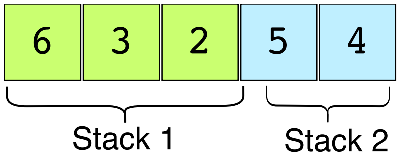
This review provides a detailed analysis of the different ways to solve the 'Implementing Two Stacks using one List' challenge.

We'll cover the following

Solution: Stacks on opposite ends

```
class TwoStacks:

    # constructor
    def __init__(self, n):
        self.size = n
        # populating 0s on all n indices of array arr
        self.arr = [0] * n
        self.top1 = -1
        self.top2 = self.size

    # Method to push an element x to stack1
    def push1(self, x):

        # There is at least one empty space for new element
        if self.top1 < self.top2 - 1:
            self.top1 += 1
            self.arr[self.top1] = x

        else:
            print("Stack Overflow ")
            exit(1)

    # Method to push an element x to stack2
    def push2(self, x):

        # There is at least one empty space for new element
        if self.top1 < self.top2 - 1:
            self.top2 -= 1
```

This implementation is space-efficient as it utilizes all of the available space. It doesn’t cause an overflow if there is any space available in the array. **The tops of the two stacks are the two extreme ends of the array.** The first stack starts from the `first` element at index `0` and the second starts from the `last` element. The first element in *stack2* is pushed at index `(len(arr)-1)`. Both stacks grow (or shrink) in the opposite direction. To check for overflow, all we need to do is check for space between the top elements of both stacks as reflected in the code.



Time Complexity#

All the operations take **constant** time because the array is being indexed and not resized.

← Back

Challenge 2: Implement Two Stacks U...

Challenge 3: Reversing First k Element...

☑ Completed

Report an Issue