Theory: Stack

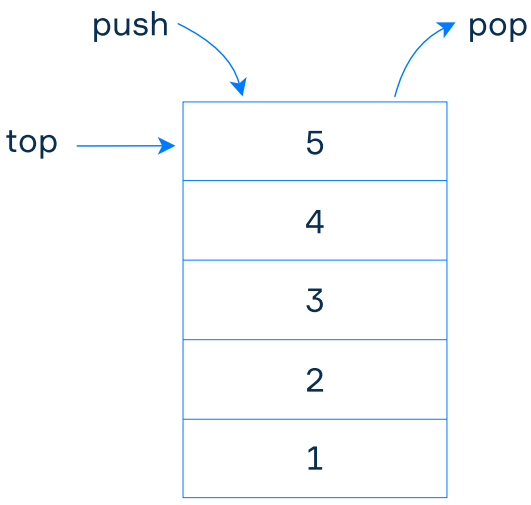
8 minutes 6 / 6 problems solved

Start practicing

§1. Stack essentials

Stack is an abstract data type where elements are inserted and removed according to the **last-in-first-out (LIFO)** principle. The **push** operation inserts an item in the top of the stack, the **pop** operation removes the top item from the stack. Access to arbitrary elements is restricted. As a rule, a stack also supports the **peek** operation that just returns the current top element. In some cases, it may also be useful to check whether the stack is empty or what is its size, so these operations should be also supported.

The following image demonstrates the basic mechanism:

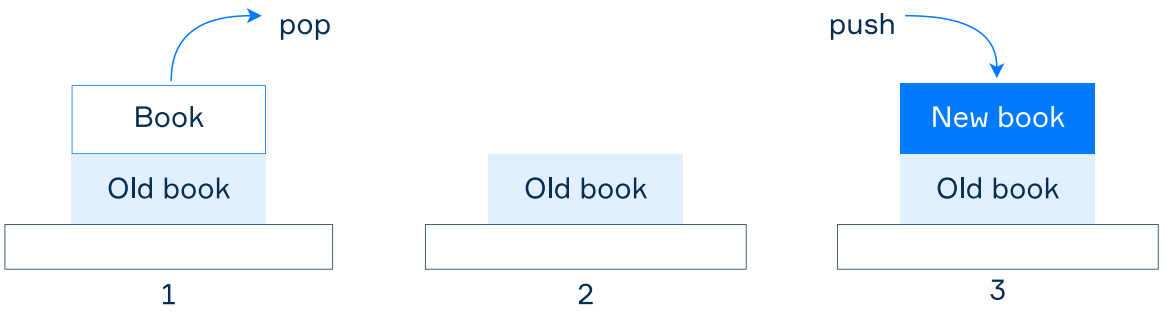


Here, element **1** was added first and will be removed last. At the same time, element **5** was added last and it's the first for removal.

The underlying data structure to implement a stack can be an array or a linked list with restricted access to its elements.

§2. Stacks in real-life and programming

The simplest real-life example is a stack of books. Only a book placed at the top can be removed at a time, but a new book is always added on the top of the stack.



You can also imagine it as a stack of plates or a pistol magazine. Also, you might have seen a StackOverflow logo before.

In programming, stacks are used to:

- evaluate arithmetic expressions;
- store arguments of functions and result of the functions' calls;
- reverse the order of elements.

§3. The efficiency of stacks

If you used a linked list or a classic array (non-resizable) as an internal structure, both **push** and **pop** operations always take constant O(1) time. It does not depend on how many elements there are in the stack, so the operations are very quick.

Report a typo

Current topic:

- ✓ Stack

Topic depends on:

- ✓ Data structures

Topic is required for:

- ✓ Deque
- ✓ Call stack
- Class files and Bytecode
- ✓ Stack in Python
- Call stack

Table of contents:

378 users liked this theory. **0** didn't like it. What about you?

Start practicing

Comments (13) Hints (0) Useful links (0) Show discussion