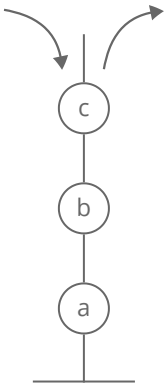

Stack

Data Structure (/data-structures-reference)

Quick reference

A **stack** stores items in a last-in, first-out (LIFO) order.

Picture a pile of dirty plates in your sink. As you add more plates, you bury the old ones further down. When you take a plate off the top to wash it, you're taking the last plate you put in. "Last in, first out."

	Worst Case
space	$O(n)$
push	$O(1)$
pop	$O(1)$
peek	$O(1)$

push(b)

Strengths:

- **Fast operations**. All stack operations take $O(1)$ time.

Uses:

- **The call stack** is a stack that tracks function calls in a program. When a function returns, which function do we "pop" back to? The last one that "pushed" a function call.
- **Depth-first search** (/concept/dfs) uses a stack (sometimes the call stack) to keep track of which nodes to visit next.
- **String parsing**—stacks turn out to be useful for several types of string parsing (/question/bracket-validator).

Implementation

You can implement a stack with either a linked list (/concept/linked-list) or a dynamic array (/concept/dynamic-array)—they both work pretty well:

	Stack Push	**Stack Pop**
Linked Lists	insert at head	remove at head
Dynamic Arrays	append	remove last element

← course home (/table-of-contents)

Next up: Largest Stack → (/question/largest-stack?course=fc1&section=queues-stacks)