Data Structures

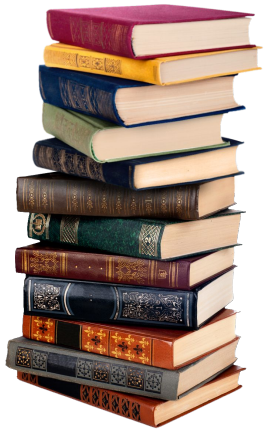
Stack Data Structure

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

What is a stack?

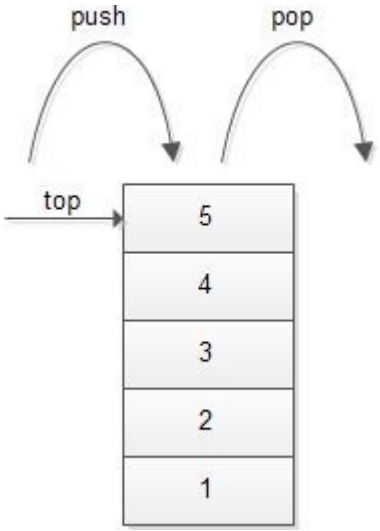
- In English, a stack denotes a pile of objects
  - A stack of boxes
  - A stack of plates
  - We **can stack** books, just like in the image
- Let's say we stacked 20 books
  - If I asked you to get a book, which one you can easily unstack (remove)?
  - The last (**top**) one?
  - When can we easily retrieve the very first book in the stack?
    Only after all others are removed
  - We call this **FILO (First in, Last out)**
    - Or Last In, First Out (LIFO)



Img src

Stack ADT

- The FILO stack is very common in practice, so we need a DS for it
- Possible functionalities? We want to add & remove elements
- **push**(element): Add to the **top** of stack
- **pop**(): Remove the **top** element in the stack
- **peek**(): Look at the 'top' element in the stack, without removing it
- Useful additional functionalities:
  - IsEmpty(): checks if the stack has any elements or not
  - IsFull(): checks if the stack has reached the maximum possible size/capacity of elements
- Any implementation that satisfies this (FILO) = Stack



Img src

Stack ADT

- Let’s trace the following operations
- The stack is initially empty!

Stack ADT

- Let’s trace the following operations
- The stack is initially empty!
- Push 5

Stack ADT

- Let’s trace the following operations
- The stack is initially empty!
- Push 5
- Push 8

Stack ADT

- Let’s trace the following operations
- The stack is initially empty!
- Push 5
- Push 8
- Push 4

Stack ADT

- Let’s trace the following operations
- The stack is initially empty!
- Push 5
- Push 8
- Push 4
- Peek? 4

Stack ADT

- Let’s trace the following operations
- The stack is initially empty!
- Push 5
- Push 8
- Push 4
- Peek? 4
- Pop

Stack ADT

- Let’s trace the following operations
- The stack is initially empty!
- Push 5
- Push 8
- Push 4
- Peek? 4
- Pop
- Push 7

Stack ADT

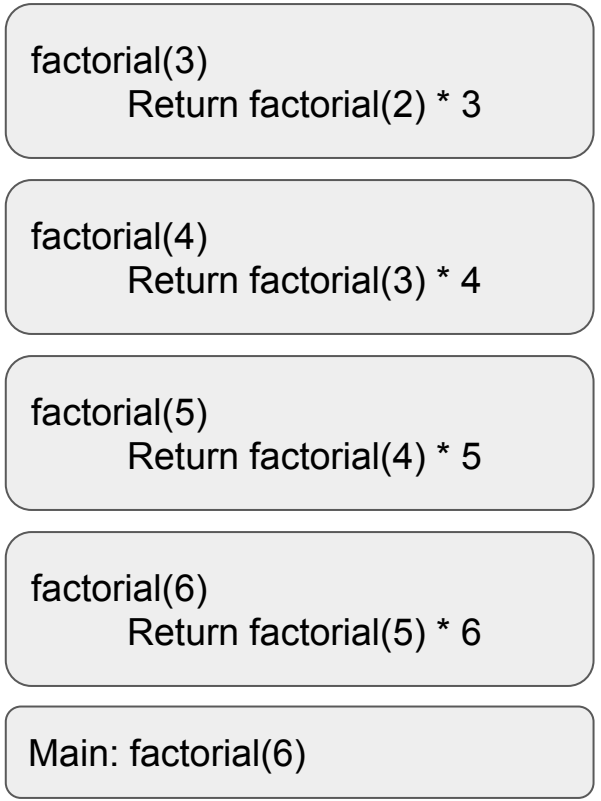
- Let’s trace the following operations
- The stack is initially empty!
- Push 5
- Push 8
- Push 4
- Peek? 4
- Pop
- Push 7
- Push 15

Stack ADT

- Let’s trace the following operations
- The stack is initially empty!
- Push 5
- Push 8
- Push 4
- Peek? 4
- Pop
- Push 7
- Push 15
- Is Empty? False

Stack: Function Calls

- When function A calls function B, which in turn calls function C, this is a stack of function calls (A, B, C) internally
  - Function C, the top, must be done first
  - Then we move back to B
  - Finally, our original call, A, is completed last
- It's similar to recursion
- For a factorial (6), we expect several calls



Stack: Tracing recursion

```
def factorial(n):
    if n <= 1:
        return 1

    return n * factorial(n-1)
```

- Call **Factorial**(6)
  - If 6 == 1? False
  - Call **Factorial** (5) and multiply the result with 6
    - If 5 == 1? False
    - Call **Factorial** (4) and multiply the result with 5
      - If 4 == 1? False
      - Call **Factorial** (3) and multiply the result with 4
        - If 3 == 1? False
        - Call **Factorial** (2) and multiply the result with 3
          - If 2 == 1? False
          - Call **Factorial** (1) and multiply the result with 2
            - If 1 == 1? True
              - Return 1

Stack Implementation

- We now know what a Stack is, as well as its ADT
- We know we have 2 memory models: Array and Linked List
- We can use either model to implement a stack
- Both implementations are simple and intuitive!
    - Try to code it yourself!

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."