بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

In the name of Allah, Most Gracious, Most Merciful

CSE 4303
Data Structure

Topic: Stacks & Its Applications

Asaduzzaman Herok
Lecturer | CSE | IUT
asaduzzaman34@iut-dhaka.edu

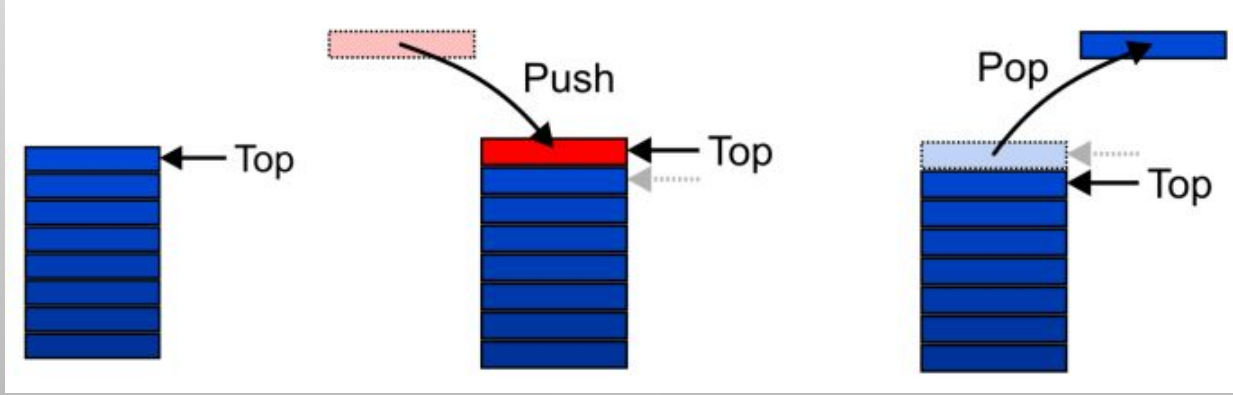
Stacks

According to the Merriam webster dictionary:

Stack(noun): a large usually conical pile (as of hay, straw, or grain in the sheaf) left standing in the field for storage

Stack(verb): to arrange in a pile or to pile in or on.

Stack(Data structure): A data structure where elements are arranged in pile and supports last-in–first-out (LIFO) behavior.



Why do need Stack

Applications of Stack:

- Parsing code:
  - Matching parenthesis
  - XML (e.g., XHTML)
- Tracking function calls
- Dealing with undo/redo operations
- Reversing a list
- Conversion of an infix expression into a postfix expression
- Evaluation of a postfix expression
- Conversion of an infix expression into a prefix expression
- Evaluation of a prefix expression
- Recursion
- … … …

Stack ADT Interface

- boolean isEmpty(); // return true if empty
- boolean isFull(); // return true if full
- void push(item); // insert item into stack
- void pop(); // remove most recent item
- void clear(); // remove all items from stack
- Item top(); Item peek() // retrieve most recent item
- Item topAndPop(); // return & remove most recent item

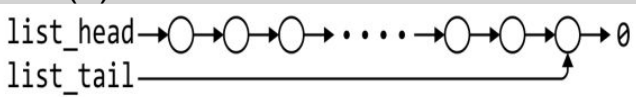
Implementations

Our target is to make asymptotic run time of any stack operation is Θ(1).

We will look at two implementations of stacks:

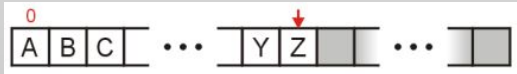
- Singly linked lists
- One-ended arrays

Operations at the front of a singly linked list are all Θ(1)



	Front/1^{st}	Back/n^{th}
Find	$\Theta(1)$	$\Theta(1)$
Insert	$\Theta(1)$	$\Theta(1)$
Erase	$\Theta(1)$	$\Theta(n)$

For one-ended arrays, all operations at the back are Θ(1)



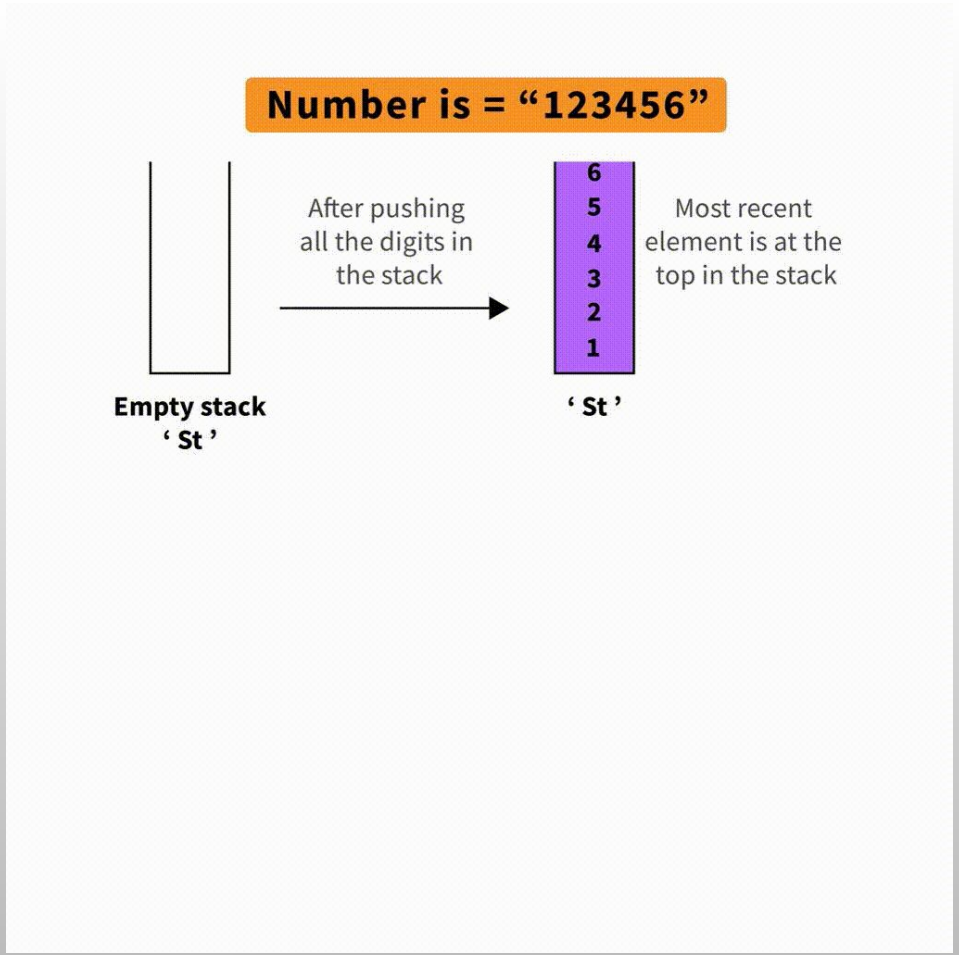
	Front/1^{st}	Back/n^{th}
Find	$\Theta(1)$	$\Theta(1)$
Insert	$\Theta(n)$	$\Theta(1)$
Erase	$\Theta(n)$	$\Theta(1)$

Implementations

- boolean isEmpty(); // return true if empty
- boolean isFull(); // return true if full
- void push(item); // insert item into stack
- void pop(); // remove most recent item
- void clear(); // remove all items from stack
- Item top(); Item peek() // retrieve most recent item
- Item topAndPop();

Follow the board Please

Application: Reversing String



Application: Parsing (XHTML)

XHTML is made of nested

- opening tags, e.g., <some_identifier>, and
- matching closing tags, e.g., </some_identifier>

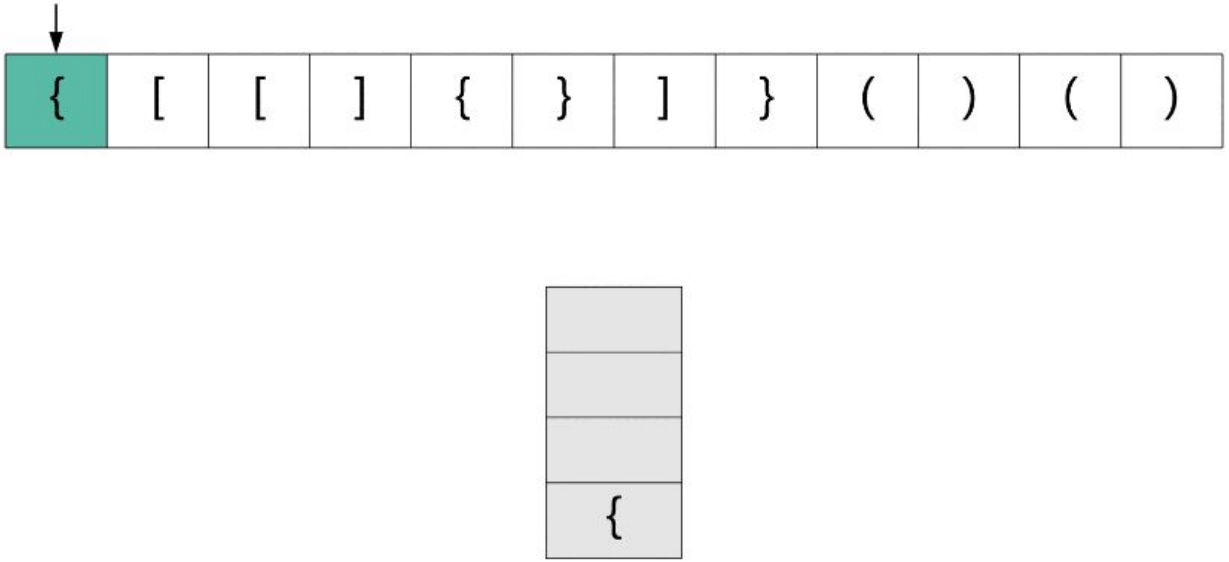
Nesting indicates that any closing tag must match the most recent opening tag

Strategy for parsing XHTML:

- read though the XHTML linearly
- place the opening tags in a stack
- when a closing tag is encountered, check that it matches what is on top of the stack

```
<html>
<head><title>Hello</title></head>
<body><p>This appears in the <i>browser</i>.</p></body>
</html>
```

Application: Matching Parenthesis



Infix, Postfix, Prefix Notation

Infix, postfix (Reverse-Polish), and prefix (Polish) notations are three different but equivalent notations of writing algebraic expressions.

Infix	Postfix	Prefix
(a + b) * c	a b + c *	* + a b c
a + (b* c)	a b c * +	+ a * b c

Infix form : <identifier> <operator> <identifier>
Postfix form : <identifier> <identifier> <operator>
Prefix form : <operator> <identifier> <identifier>

Application: Evaluate Postfix Notation

Infix notation: **9 – ((3 * 4) + 8) / 4**
Postfix notation: **9 3 4 * 8 + 4 / –**

Character Scanned	Stack
9	9
3	9, 3
4	9, 3, 4
*	9, 12
8	9, 12, 8
+	9, 20
4	9, 20, 4
/	9, 5
–	4

```
Step 1: Add a ")" at the end of the
        postfix expression
Step 2: Scan every character of the
        postfix expression and repeat
        Steps 3 and 4 until ")"is encountered
Step 3: IF an operand is encountered,
        push it on the stack
        IF an operator O is encountered, then
        a. Pop the top two elements from the
           stack as A and B as A and B
        b. Evaluate B O A, where A is the
           topmost element and B
           is the element below A.
        c. Push the result of evaluation
           on the stack
        [END OF IF]
Step 4: SET RESULT equal to the topmost element
        of the stack
Step 5: EXIT
```

Application: Expressions into Postfix Expressions

Infix: $Q = A + B * C - D / E \uparrow F * G * H$

Postfix: P = ABC * + DEF ↑ / G * H * -

```
Step 1: Add ")" to the end of the infix expression
Step 2: Push "(" on to the stack
Step 3: Repeat until each character in the infix notation is scanned
       IF a "(" is encountered, push it on the stack
       IF an operand (whether a digit or a character) is encountered, add it to the
       postfix expression.
       IF a ")" is encountered, then
         a. Repeatedly pop from stack and add it to the postfix expression until a
            "(" is encountered.
         b. Discard the "(". That is, remove the "(" from stack and do not
            add it to the postfix expression
       IF an operator O is encountered, then
         a. Repeatedly pop from stack and add each operator (popped from the stack) to the
            postfix expression which has the same precedence or a higher precedence than O
         b. Push the operator O to the stack
       [END OF IF]
Step 4: Repeatedly pop from the stack and add it to the postfix expression until the stack is empty
Step 5: EXIT
```

Acknowledgement

Rafsanjany Kushol
PhD Student, Dept. of Computing Science,
University of Alberta

Sabbir Ahmed
Assistant Professor
Department of CSE, IUT