Data Structures

lecture 15

4-11-2022

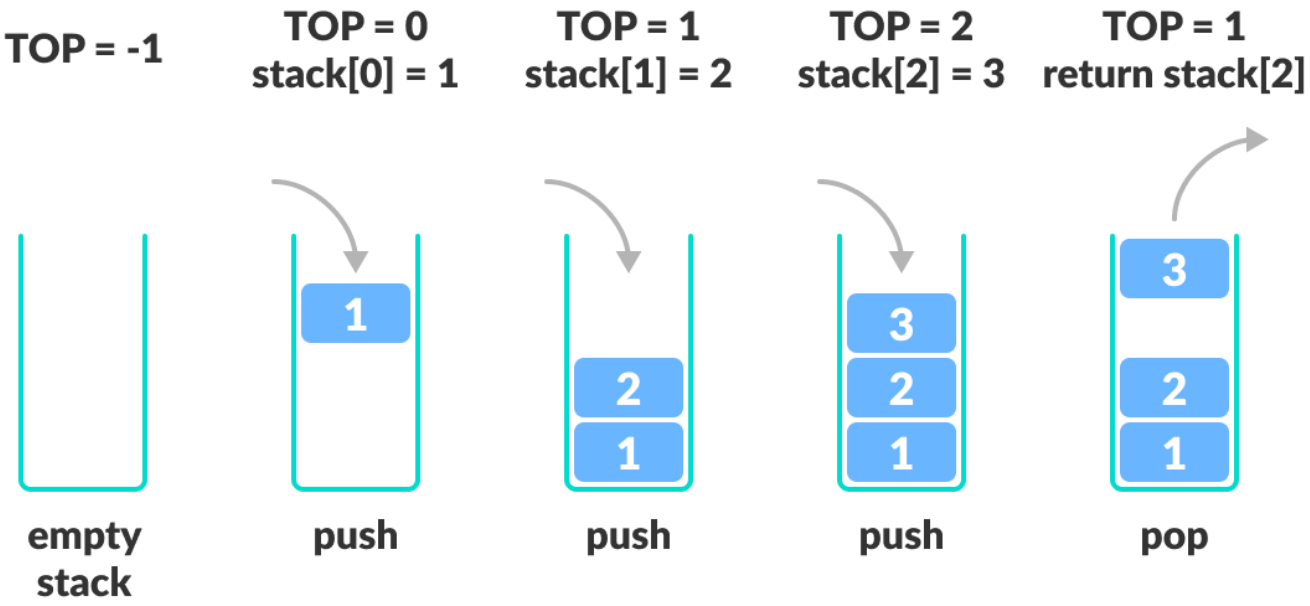
Unit 3: Stack and Queue

Stack

Stack

- Simple data structure allows adding and removing elements in a particular order **(LIFO).**
- only one open end called as **Top** from where elements are added or removed.

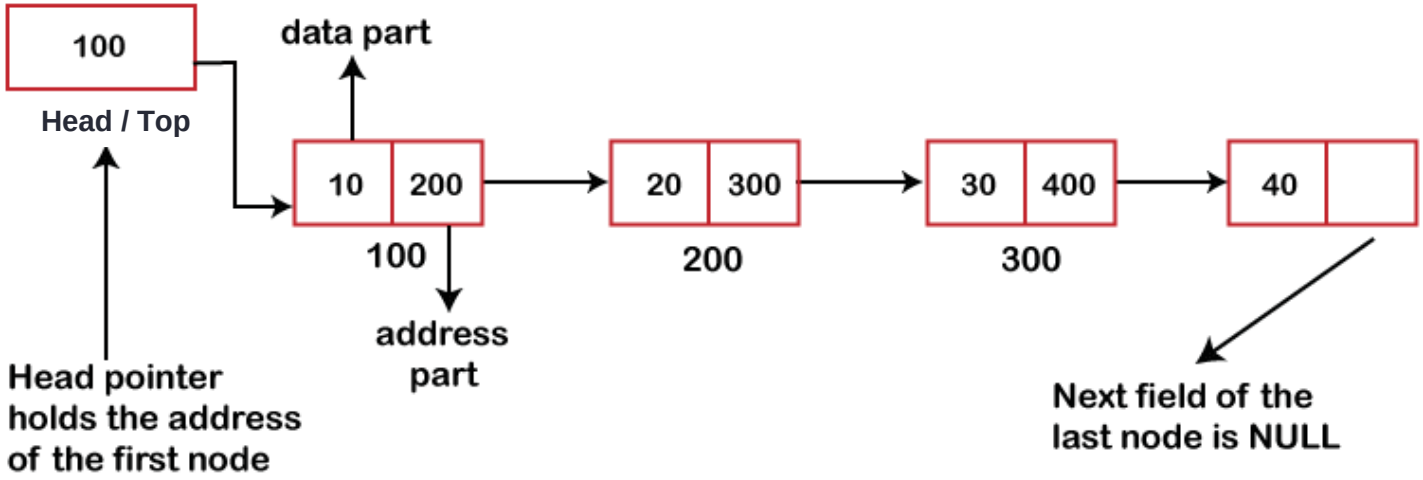
Basic Operations on Stack



Implementing Stack as an Array

- A stack can be represented as an array
- the capacity of array (n) => max capacity of stack.
- Since the array index start from zero
  - top can go up to n-1 (n = capacity of array)

Implementing Stack as List (Linked List)



Applications of Stack

- Reversing of data (String/word)
- Conversion from **decimal to binary**
- **Navigation** in **web browsers** (maintain history of WebPages) and in **mobile applications**.
- **Expression Handling :** highly effective in evaluating arithmetic operations by converting it from one form to other
  - **example :** converting infix expression to postfix

Reversing of data (String/word)

- Word: “kitcoek”
- Reverse: “keoctik”

Conversion from decimal to binary

- 22 ➔ ?

Navigations in Browsers and Mobile Apps

- 22 ➔ ?

Expression Handling / Conversion

- 3 Types of Expression
  - Infix expression
    - A + B * C - D / E
  - Postfix expression
    - A B C * + D E / -
  - Prefix expression
    - +A -* B C / D E

Converting infix expression to postfix expression

Step 1 : Scan the Infix Expression from left to right.

Step 2 : If the scanned character is an operand, append it with final Infix to Postfix string.

Step 3 : Else,

Step 3.1 : If the precedence order of the scanned(incoming) operator is greater than the precedence order of the operator in the stack (or the stack is empty or the stack contains a ‘(‘ or ‘[‘ or ‘{‘), push it on stack.

Step 3.2 : Else, Pop all the operators from the stack which are greater than or equal to in precedence than that of the scanned operator. After doing that Push the scanned operator to the stack. (If you encounter parenthesis while popping then stop there and push the scanned operator in the stack.)

Step 4 : If the scanned character is an ‘(‘ or ‘[‘ or ‘{‘, push it to the stack.

Step 5 : If the scanned character is an ‘)’or ‘]’ or ‘}’, pop the stack and and output it until a ‘(‘ or ‘[‘ or ‘{‘ respectively is encountered, and discard both the parenthesis.

Step 6 : Repeat steps 2-6 until infix expression is scanned.

Step 7 : Print the output

Step 8 : Pop and output from the stack until it is not empty.