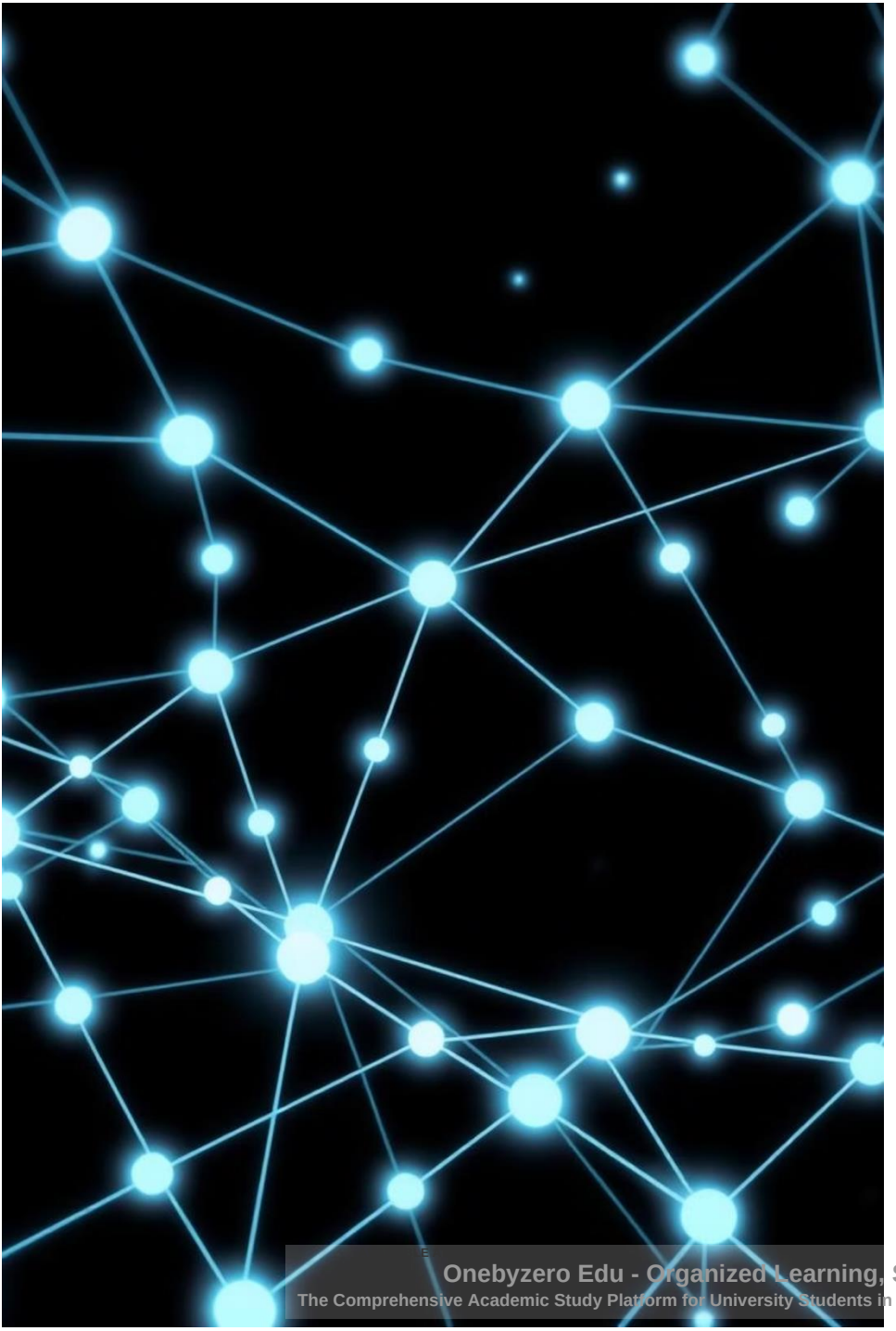


Data Structures

Lecture 5: Stack

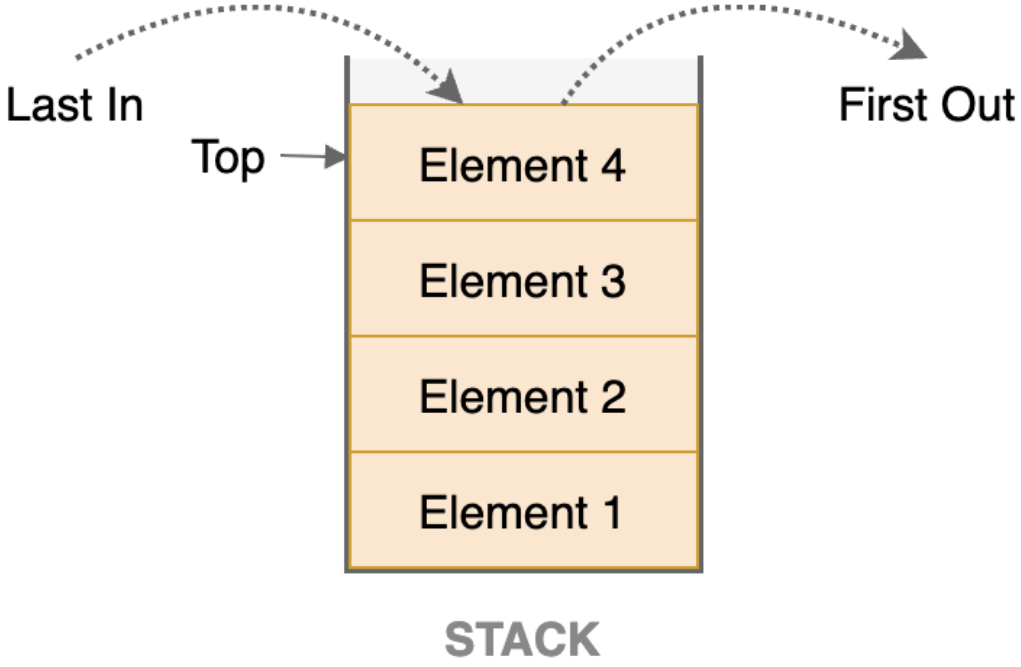
Instructor:
Md Samsuddoha
Assistant Professor
Dept of CSE, BU

Contents

- Concept of Stack
- Complexity of Stack
- Coding Stack
- Problem Solving using Stack

Stacks: Last In, First Out (LIFO) Collections

- **The Plate Analogy**
  - Imagine a stack of plates: you can only add a new plate to the top, and you can only take the top plate off. This "Last In, First Out" (LIFO) principle defines stack behavior.
- **Key Operations**
  - **Push:** Adds an element to the top of the stack.
  - **Pop:** Removes the top element from the stack.
  - **Peek:** Views the top element without removing it.
- **Ubiquitous Applications**
  - Stacks are critical for managing function calls in programming, implementing "undo" mechanisms in software, and evaluating mathematical expressions. They can be efficiently built using either arrays or linked lists.



Properties of Stack

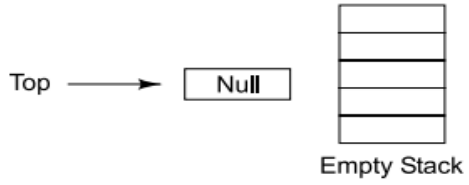
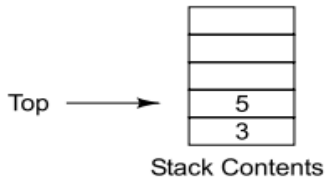
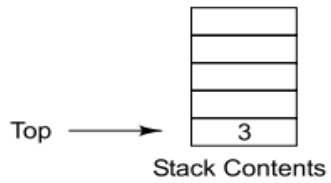
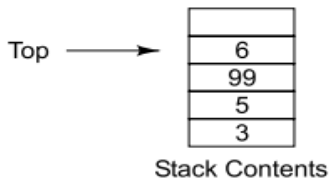
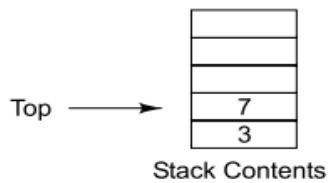
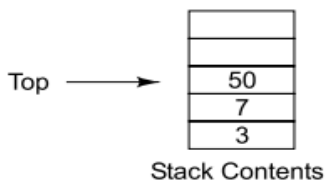
- **Top**: The top of the stack
- **Element**: The actual data
- **Push**: A new element is inserted on Top of the stack
- **Pop**: Element is removed from the Top of the stack
- **Underflow**: Stack is empty, but requested for pop
- **Overflow**: Stack reaches its capacity, but requested for push (applicable only for the array-based implementation of the stack)

Real Life Application of Queues

- **Undo/Redo in Editors**
  - Example: MS Word, Photoshop
  - Most recent action is undone first → **LIFO**
- **Browser Back Button**
  - When navigating webpages, previous pages are pushed to stack.
  - Pressing "Back" pops the last visited page.
- **Call Stack in Programming**
  - Tracks function calls.
  - When a function is called, it's pushed to the stack.
  - When it finishes, it's popped.
- **Expression Evaluation & Syntax Parsing:** Used in compilers to evaluate expressions like ((a+b)*c).
- **Reversing Text:** Pushing characters to a stack and popping them gives reversed order. (e.g abc→cba)

Stack Operations (LIFO)

- **Push**: Insert new element at top
- **Pop**: remove top element

Initial Empty Stack	
Push (3), Push (5)	
Pop ()	
Push(7), Push (99), Push (6)	
Pop (), Pop ()	
Push (50)	
Pop (), Pop (), Pop ()	

```
STACK-EMPTY(S)
1  if S.top == 0
2       return TRUE
3  else return FALSE

PUSH(S, x)
1  S.top = S.top + 1
2  S[S.top] = x

POP(S)
1  if STACK-EMPTY(S)
2       error "underflow"
3  else S.top = S.top - 1
4       return S[S.top + 1]
```

Applications of Stack

- Expression Evaluation and Conversion
  - Evaluating Postfix (34*5+)
  - Converting Infix to postfix
  - Evaluating Infix
- Balanced Parenthesis Checking
- Undo/Redo in text editors
- String Reversal
- Tower of Hanoi
- Depth First Search (DFS)

References

- **Chapter 6:**
  - **Data Structures using C** by E. Balagurusamy

- **Chapter 10**: Introduction to Algorithms (Cormen)

Thank You