Limitations of Static Stack

📌 **Limited Flexibility**

- The stack size must be **declared at design time** and **cannot be changed** later.

📌 **Inefficient Memory Utilization**

- **Underutilization**: If the stack holds **fewer elements** than allocated, memory is **wasted**.
- **Overflow Risk**: If more elements are needed, the array **cannot expand**, leading to **stack overflow**.

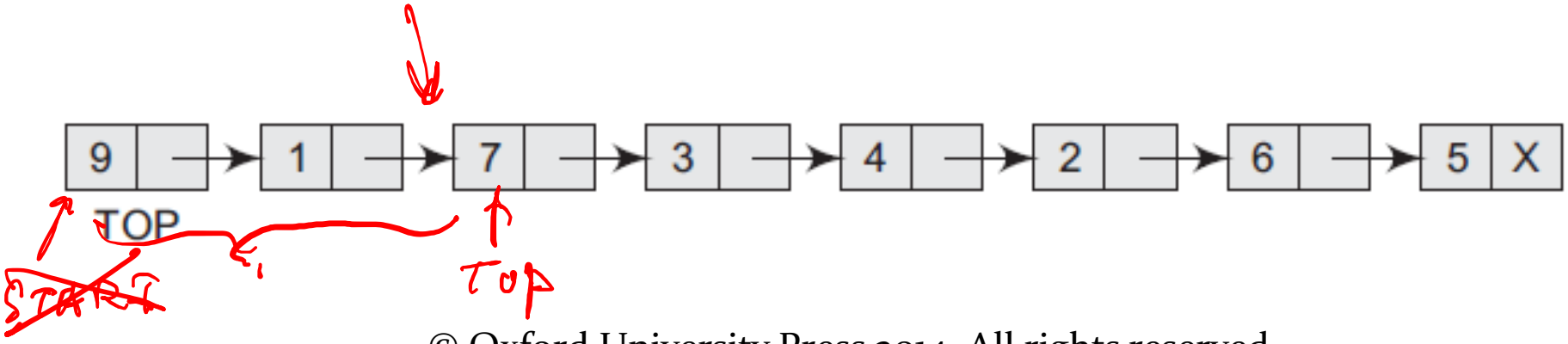
📌 **Fixed Size Constraint**

- Once defined, the **stack size remains constant**, making it **unsuitable for dynamic applications**.

💡 **Solution**: **Dynamic Stack Implementation** (Using linked lists)

Linear Representation of Stacks

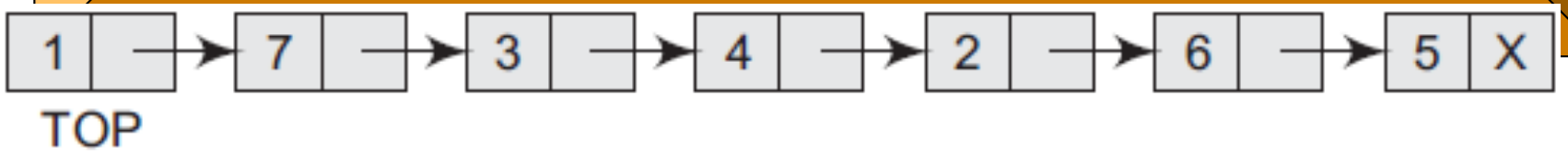
- In a linked stack, every node has two parts – one that stores data and another that stores the address of the next node.
- The START pointer of the linked list is used as TOP.
- If TOP is NULL, then it indicates that the stack is empty.
- The stack does not have a maximum limit other than the amount of memory that can be dynamically allocated.

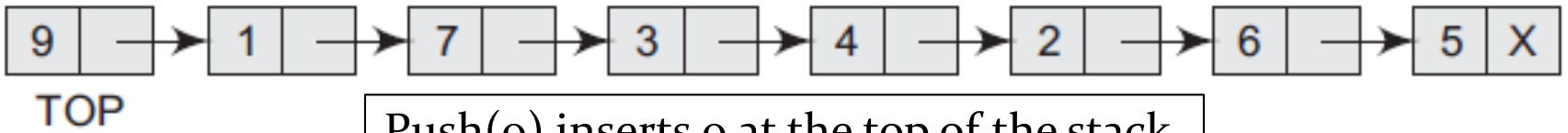


Push Operation on a Linked Stack

```
Algorithm to PUSH an element in a linked stack

Step 1: Allocate memory for the new node as New_Node
Step 2: SET New_Node->DATA = VAL
Step 3: IF TOP = NULL, then
              SET New_Node->NEXT = NULL
              SET TOP = New_Node
        ELSE
              SET New_node->NEXT = TOP
              SET TOP = New_Node
        [END OF IF]
Step 4: END
```



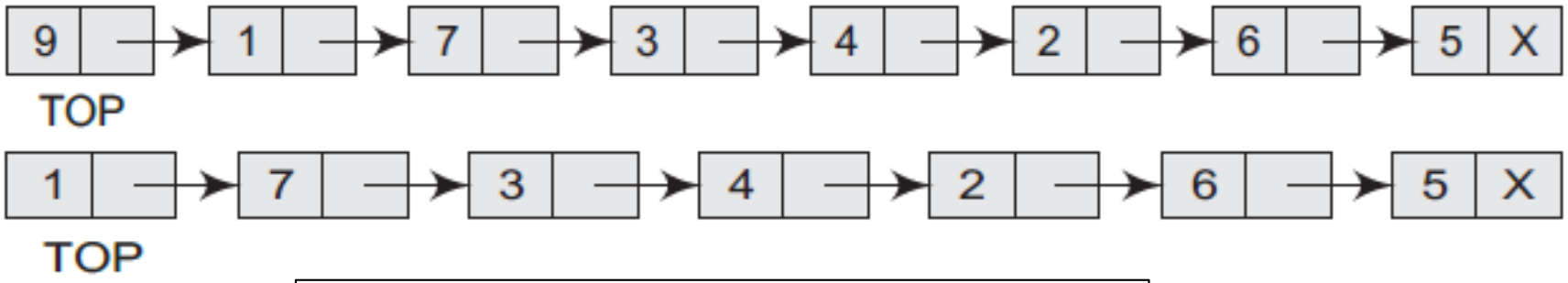


Push(9) inserts 9 at the top of the stack

Pop Operation on a Linked Stack

```
Algorithm to POP an element from a stack

Step 1: IF TOP = NULL, then
                PRINT "UNDERFLOW"
                GOTO Step 6
        [END OF IF]
Step 2: SET PTR = TOP
Step 3: SET TOP = TOP ->NEXT
Step 4: VAL = PTR->DATA
Step 5: FREE PTR
Step 6: END
```



Pop() deletes 9 from the top of the stack

Peek Operation on a Linked Stack

```
Algorithm to PEEK at the top stack element

Step 1: IF TOP = NULL, then
            PRINT "STACK IS EMPTY"
            GOTO Step 3
        [END OF IF]
Step 2: RETURN TOP->DATA
Step 3: END
```

Peek() returns 9

Underflow

Linked list Top == NULL?

Array Top == -1 ?

Overflow

AVAIL == NULL?

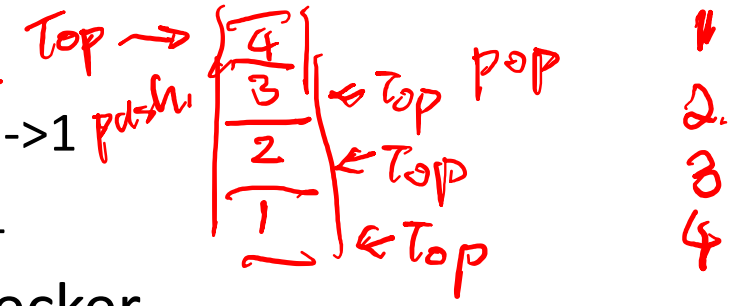
Top == MAX - 1 ?

Time Complexity – Worst Case

PUSH	O(1)
POP	O(1)
Peek	O(1)

Applications of Stacks

- Reversing a list
  - A list of numbers can be reversed by pushing each number from the first position to the last position onto a stack and then popping each number off the stack starting with the first position in the reversed list.
  - List: 1, 2, 3, 4
  - Stack: 4->3->2->1
  - RList: 4, 3, 2, 1



- Parentheses checker
  - Stacks can be used to push open parentheses or braces and pop them as closing parentheses/braces are encountered.
  - If mismatches occur or any leftover parentheses in the stack or expression, then the parentheses or braces would be incorrect.
  - Expression: (A+B}, Stack: (, Error when popping (on }, invalid
  - Expression: {A+(B-C)}, Stack: {(, pop (matches), pop { matches }, valid