Data Structures-2

Prof. K. Adisesha

BE, M.Sc., M.Th., NET, (Ph.D.)

Learning objectives

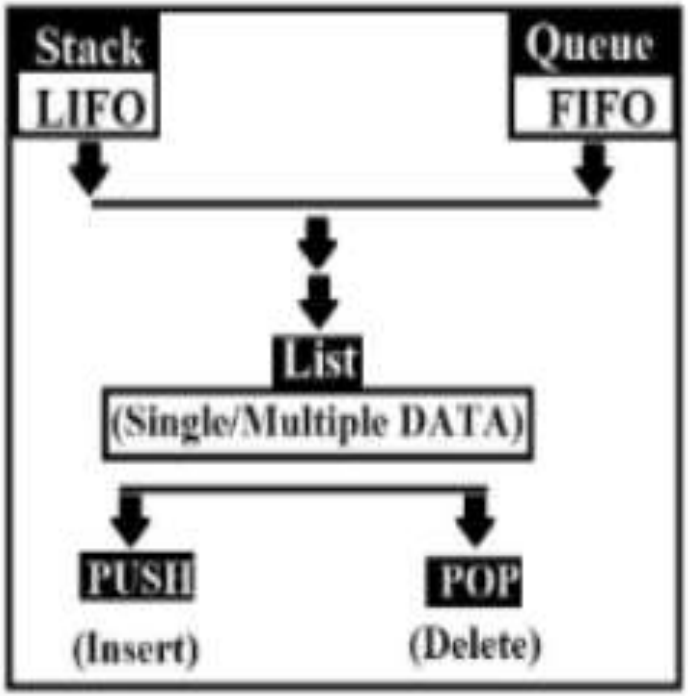
Objective

- Linear data structures
- Type Linear data structures
  - Stacks – PUSH, POP using a list (Single Data, Multiple Data)
  - Queue- INSERT, DELETE using a list (Single Data, Multiple Data)
- Implementation in Python

Data Structure

What is Data Structure ?

- Data structures are a way of organizing and storing data so that they can be accessed and worked with efficiently.
- Linear Data Structures
  - Data collections of ordered items
  - Order depends on how items are added and removed
  - Once added item stays in position
  - Examples: **stacks, queues**



Linear Data Structure

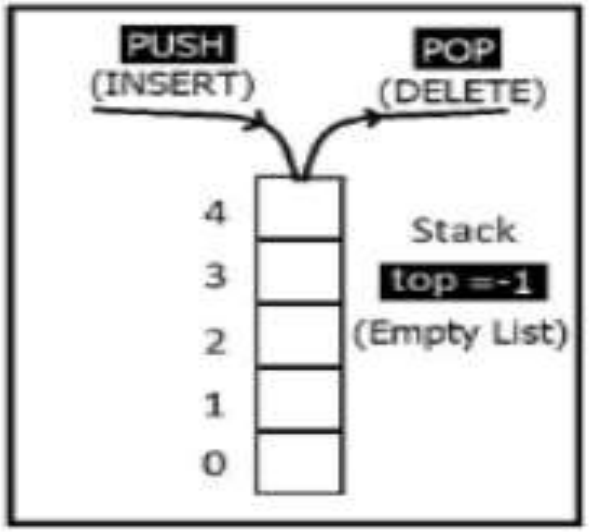
Characteristics Linear Data Structures:

- Two ends (left – right, front – rear)
- Linear structures distinguished by how items are added and removed
- Additions of new items only allowed at the end
- Deletion of existing items only allowed at the end
- Appear in many algorithms

Linear Data Structure

Stack:

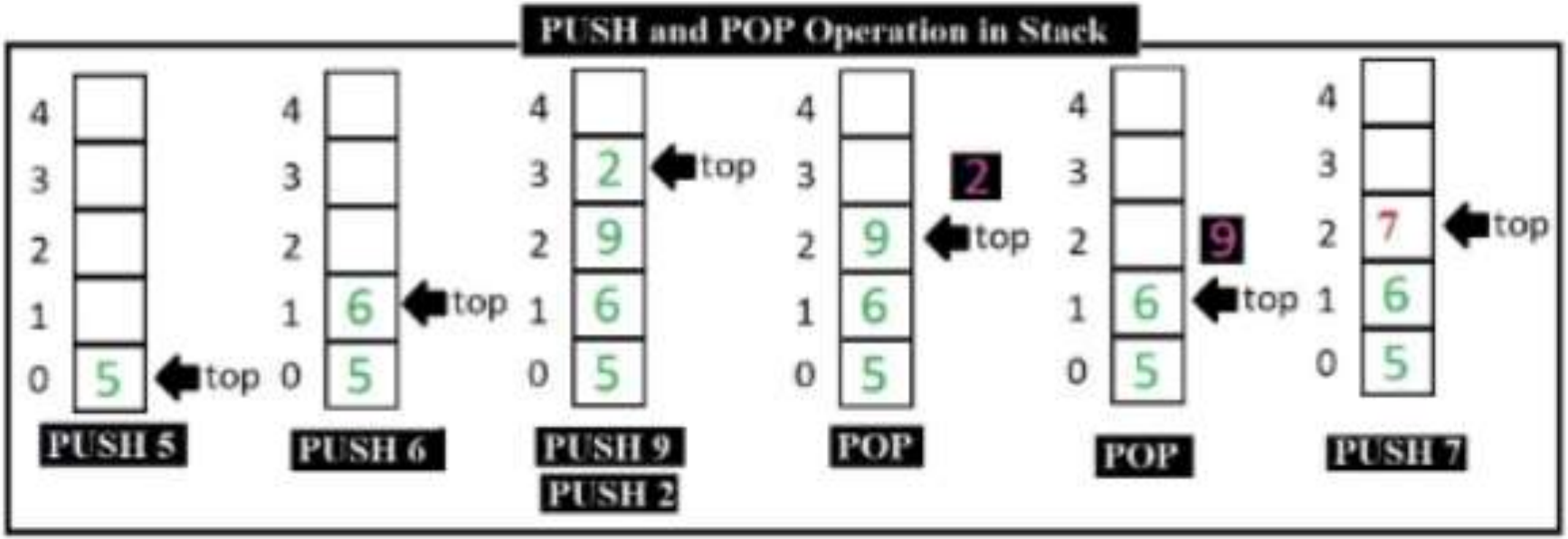
- Stacks are linear Data Structures which are based on the principle of Last-In-First-Out (LIFO) where data which is entered last will be the first to get accessed.
  - Addition/removal of items always takes place at same end (top)
  - Base represents bottom and contains item has been in stack the longest
  - Most recently added to be removed first (last in-first-out, LIFO)
  - Top: newer items;
  - bottom: lower items



Stack

Operations of Stack

- **PUSH**: pushing (adding) elements into Top of Stack,
- **POP**: Popping (deleting) elements and accessing elements from Top of Stack.
- **TOP**: This TOP is the pointer to the current position of the stack.



Stack

Applications Using Stacks

- Stacks are prominently used in applications such as:
  - Recursive Programming
  - Reversing words
  - Expression Conversion
    - In-fix to Post-fix
  - Backtracking
  - Undo mechanisms in word editors
  - Check if delimiters are matched
    - Matching of opening and closing symbols: {,},[,],(,)
    - Check: {{a}[b]{[{c}](d(e)f)}((g))} and ({[a}b(c)])

Stack

Stack - Abstract Data Type

- **Stack()** creates a new, empty stack; no parameters and returns an empty stack.
- **push(item)** adds a new item at top of stack; needs the item and returns nothing.
- **pop()** removes top item; needs no parameters, returns item, stack is modified
- **peek()** returns top item from the stack but doesn't remove it; needs no parameters, stack is not modified
- **isEmpty()** test if stack is empty; needs no parameters, returns a boolean value
- **size()** returns number of items on stack; needs no parameters; returns an integer

Stack

Implementing Stack using List in Python

```
class StackList:
    def __init__(self):
        self.items = []

    def isEmpty(self):
        return self.items == []

    def push(self, item):
        self.items.insert(0,item)

    def pop(self):
        return self.items.pop(0)

    def peek(self):
        return self.items[0]

    def size(self):
        return len(self.items)
```

Stack

Implementing Stack using List in Python

```
#Program to implement Stack Operation on Single data
def push(stack,x):      #function to add element at the end of list
    stack.append(x)
def pop(stack):         #function to remove last element from list
    n = len(stack)
    if(n<=0):
        print("Stack empty....Pop not possible")
    else:
        stack.pop()
def display(stack):     #function to display stack entry
    if len(stack)<=0:
        print("Stack empty.....Nothing to display")
    for i in stack:
        print(i,end=" ")
```

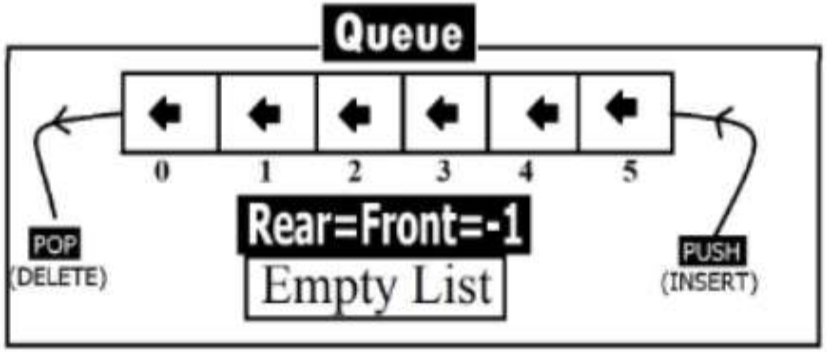
Stack

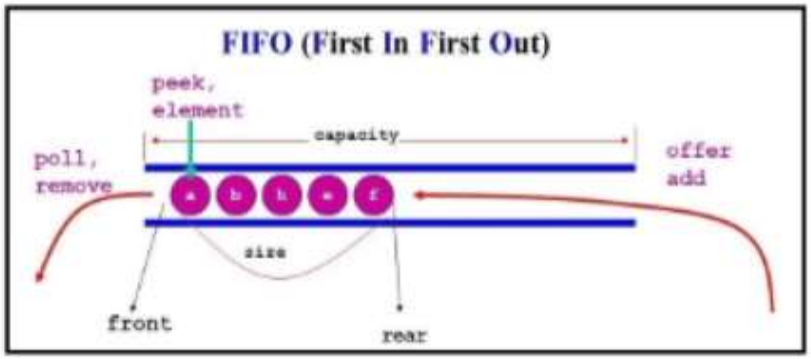
```
#main program starts from here
x=[]
choice=0
while (choice!=4):
    print("********Stack Menu**********")
    print("1. push(INSERT)")
    print("2. pop(DELETE)")
    print("3. Display ")
    print("4. Exit")
    choice = int(input("Enter your choice :"))
    if(choice==1):
        value = int(input("Enter value "))
        push(x,value)
    if(choice==2):
        pop(x)
    if(choice==3):
        display(x)
    if(choice==4):
        print("You selected to close this program")
```

Queue

Queue

- A queue is also a linear data structure which is based on the principle of First-In-First-Out (FIFO)
- where the data entered first will be accessed first.
- It has operations which can be performed from both ends of the Queue, that is, head-tail or front-back.
  - En-Queue: Add items on one end
  - De-Queue: Remove items on the other end

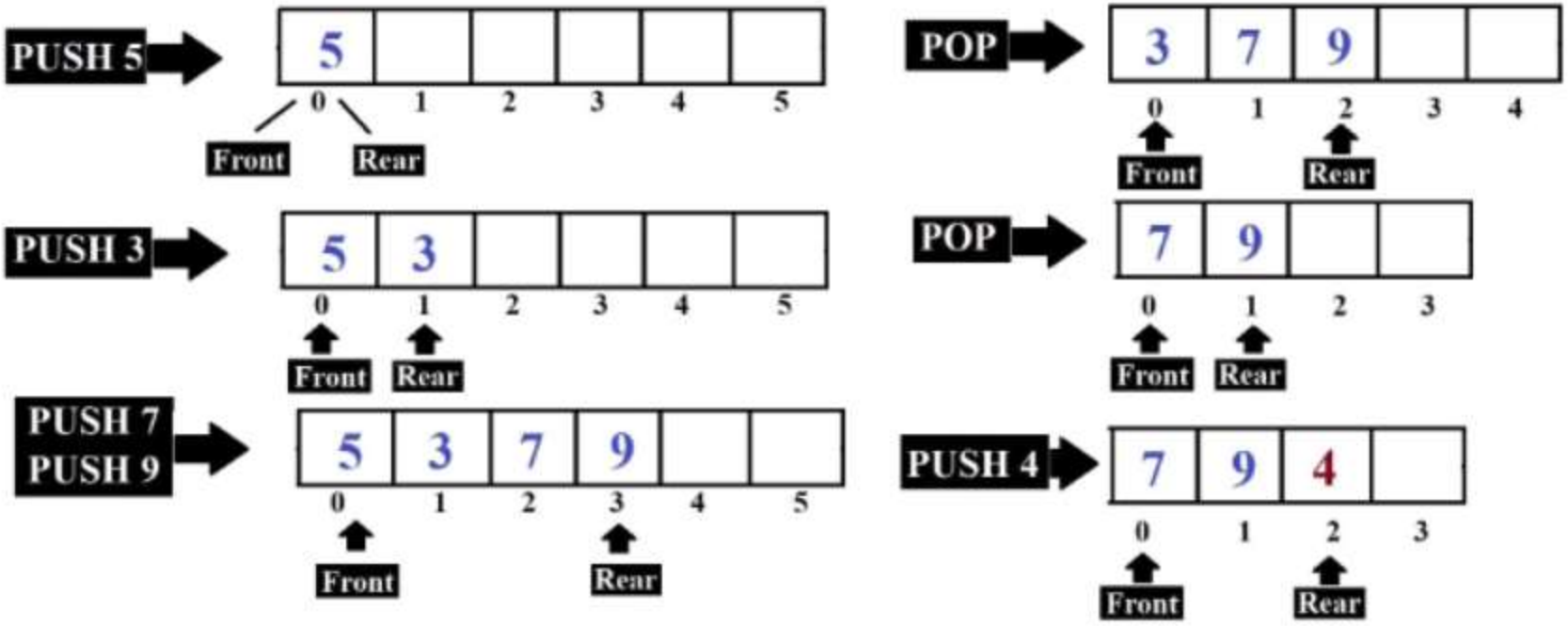




Queue

Queue

- A queue is also a linear data structure which is based on the principle of First-In-First-Out (FIFO)
  - En-Queue: Add items on one end (Rear)
  - De-Queue: Remove items on the other end (Front)



Queue

Applications of Queues

- Queues are used in various applications:
  - In network equipment like switches and routers
  - Network Buffers for traffic congestion management
  - Operating Systems for Job Scheduling
  - Checkout line
  - Printer queue
  - Take-off at airport

Queue

Queue - Abstract Data Type

- **Queue()** creates a new, empty queue; no parameters and returns an empty queue.
- **enqueue(item)** adds a new item to rear of queue; needs the item and returns nothing.
- **dequeue()** removes front item; needs no parameters, returns item, queue is modified
- **isEmpty()** test if queue is empty; needs no parameters, returns a boolean value
- **size()** returns number of items in the queue; needs no parameters; returns an integer

Queue

Implementing Queue in Python

```
class Queue:
    def __init__(self):
        self.items = []

    def isEmpty(self):
        return self.items == []

    def enqueue(self, item):
        self.items.insert(0,item)

    def dequeue(self):
        return self.items.pop()

    def size(self):
        return len(self.items)
```

```
from Queue import Queue

q=Queue()

q.enqueue(4)
q.enqueue('dog')
q.enqueue(True)
print(q.size())
```

Queue

Implementing Queue using List in Python

```
def add_element(Queue,x):   #function to add element at the end of list
    Queue.append(x)
def delete_element(Queue):  #function to remove last element from list
    n = len(Queue)
    if(n<=0):
        print("Queue empty....Deletion not possible")
    else:
        del(Queue[0])
def display(Queue):         #function to display Queue entry
    if len(Queue)<=0:
        print("Queue empty.....Nothing to display")
    for i in Queue:
        print(i,end=" ")
```

Queue

```
#main program starts from here
x=[]
choice=0
while (choice!=4):
    print(" ********Queue menu***********")
    print("1. Add Element ")
    print("2. Delete Element")
    print("3. Display ")
    print("4. Exit")
    choice = int(input("Enter your choice : "))
    if(choice==1):
        value = int(input("Enter value : "))
        add_element(x,value)
    if(choice==2):
        delete_element(x)
    if(choice==3):
        display(x)
    if(choice==4):
        print("You selected to close this program")
```

Conclusion!

- We learned about:
  - Linear data structures
  - Type Linear data structures
    - Stacks – PUSH, POP using a list (Single Data, Multiple Data)
    - Queue- INSERT, DELETE using a list (Single Data, Multiple Data)
  - Implementation in Python

Thank you