TECHNO INDIA UNIVERSITY
WESTBENGAL

Data Structures & Algorithms
(PCC-CS 301)

Dr. Debashis Das
Associate Professor
Department of CSE
Techno India University, Kolkata

Topics Covered

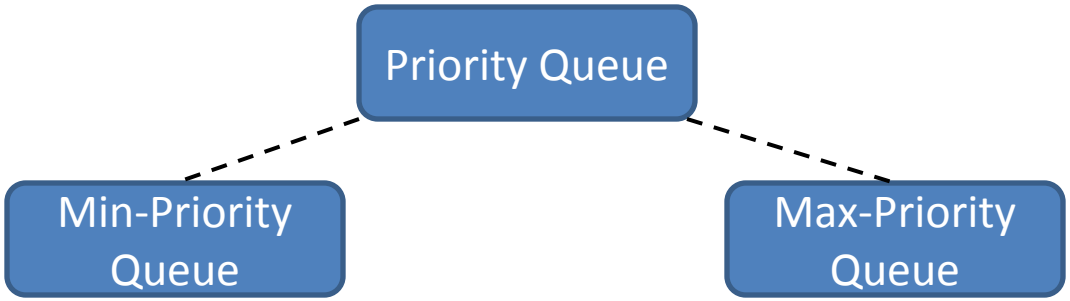
1. Linear Data Structure
   a. Priority Queue
   b. Implementation of Queue using Stack
   c. Implementation of Stack using Queue

Queue

- Priority Queue
  - Properties
    - Each element of the queue will be assigned with a priority
    - An element is processed based on its priority value
      - Higher priority data is processed before the less priority data
    - Priorities are fixed based on various applications
      - Minimum value sometimes considered as highest priority
      - In few applications, maximum data value is regarded as the highest priority
    - Priority queue does not follow **F**irst **I**n **F**irst **O**ut concept

Queue

- Priority Queue



- Min-priority Queue
  - Element with minimum priority is considered as highest priority element and that to be processed first
- Max-priority Queue
  - Element with maximum priority is considered as highest priority element and that to be processed first

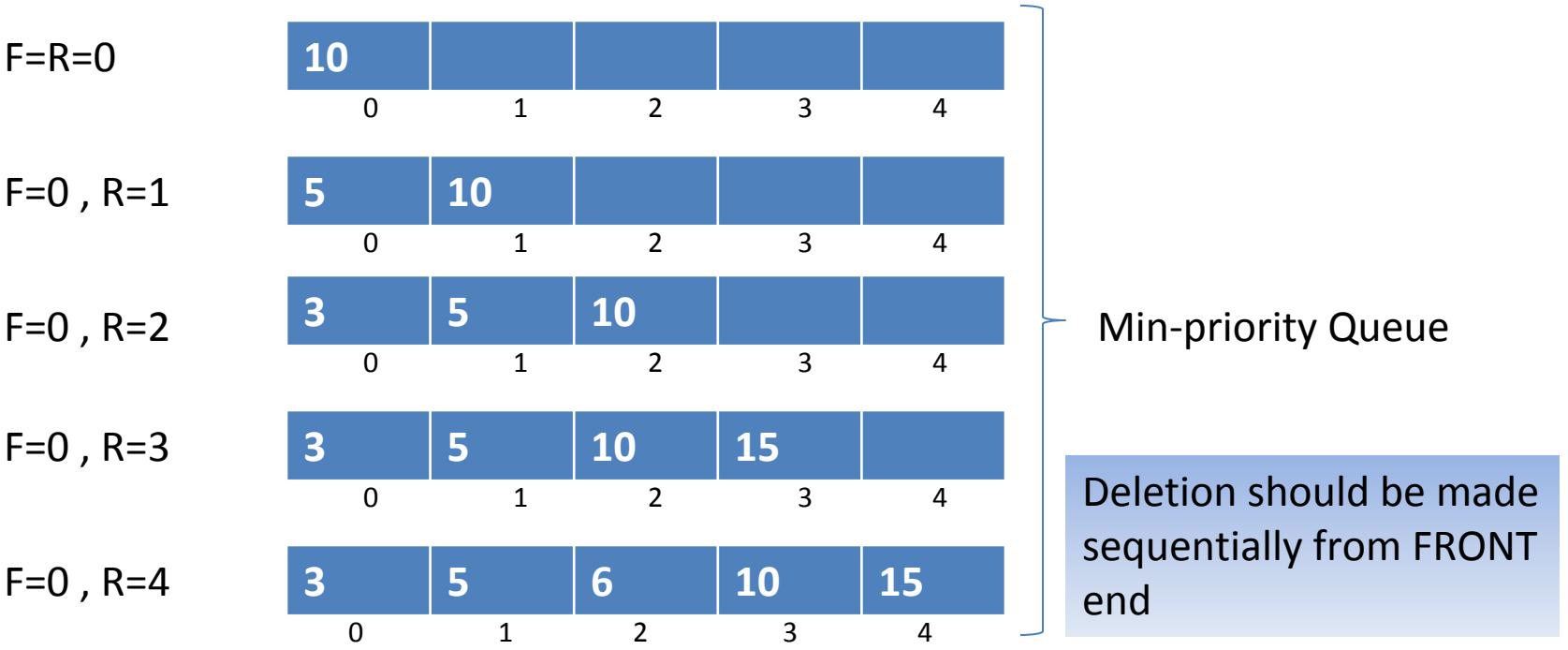
Queue

- Priority Queue
  - Representation
    - Using array
    - Using Heap (non-linear data structure, to be discussed later)
  - Array representation
    - Data can be inserted in a sequence of its priority so that it can be processed (deleted) sequentially from the start index
      - Enqueue requires O(n) and Dequeue requires O(1)
    - Data can be inserted sequentially as it encounters whereas data processing (deletion) is performed by searching the highest priority data present in the list
      - Enqueue requires O(1) and Dequeue requires O(n)

Priority Queue

- Array representation
  - Type- I

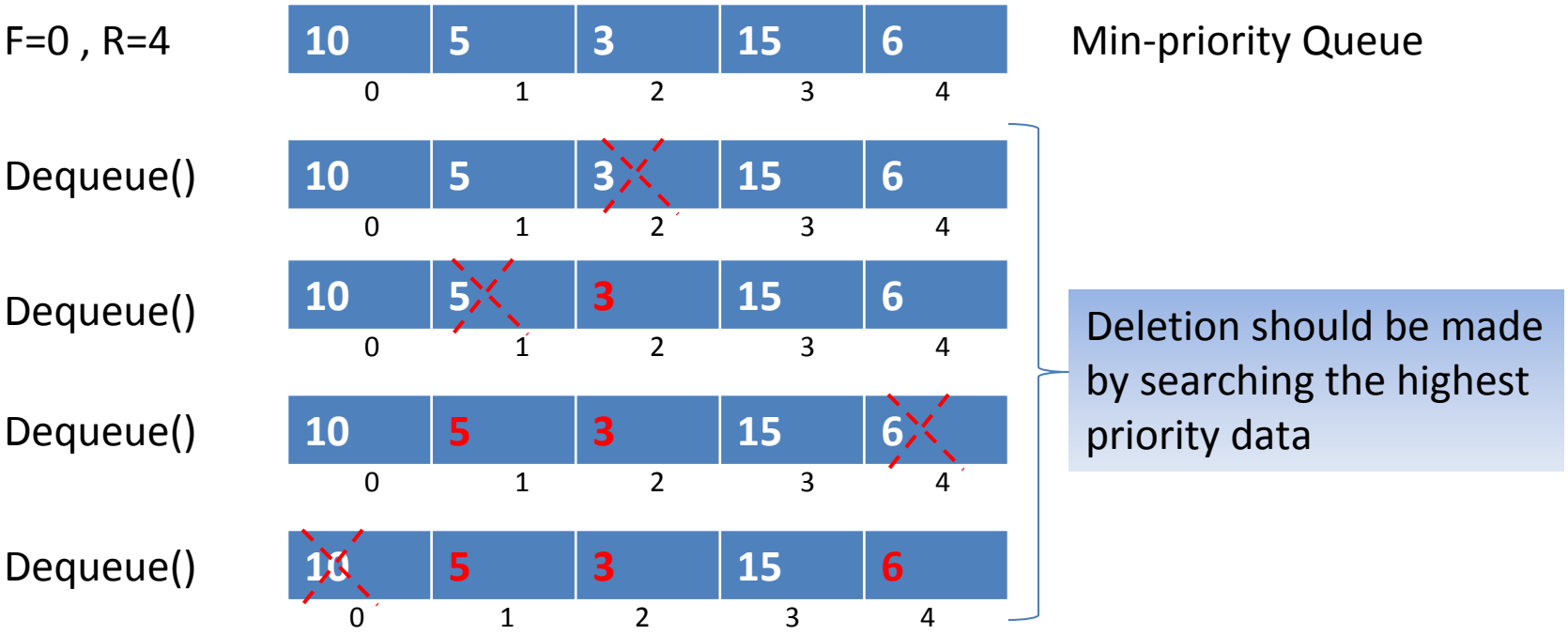
Input data: 10, 5, 3, 15, 6



Priority Queue

- Array representation
  - Type- II

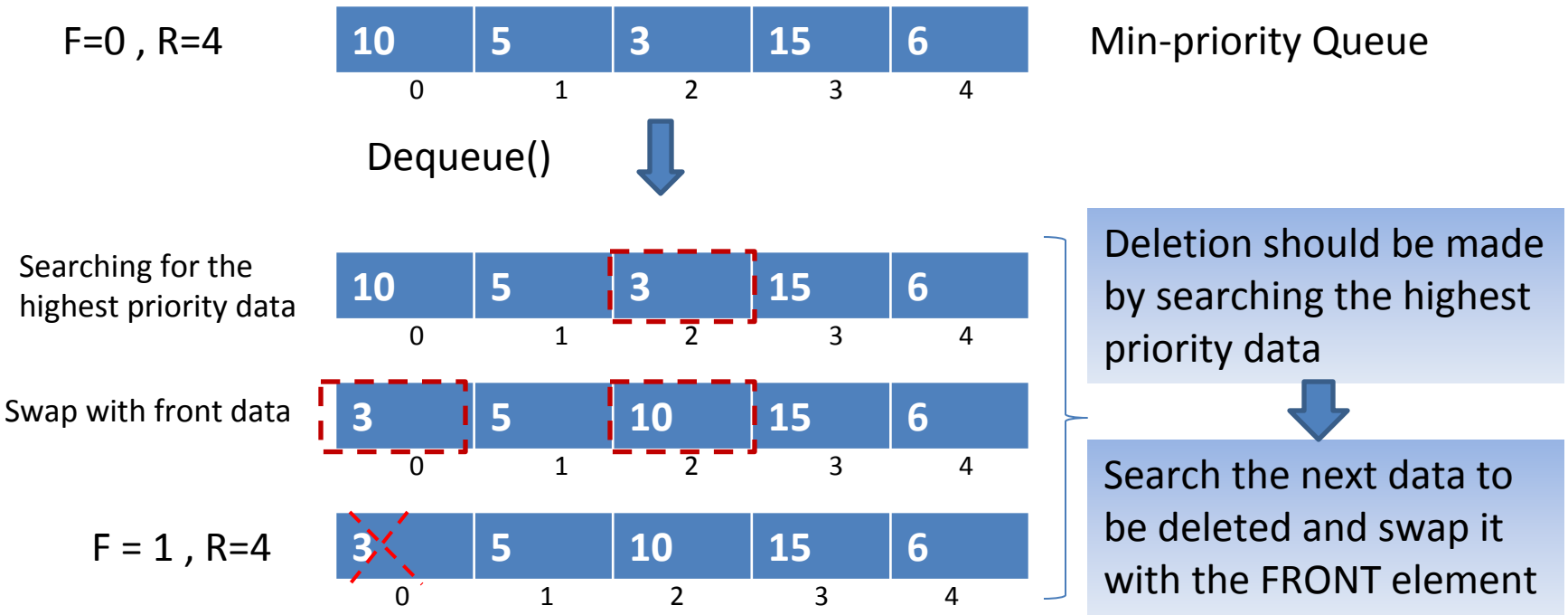
Input data: 10, 5, 3, 15, 6



Priority Queue

- Array representation
  - Type- II (implementation issue)

Input data: 10, 5, 3, 15, 6



Queue using Stack

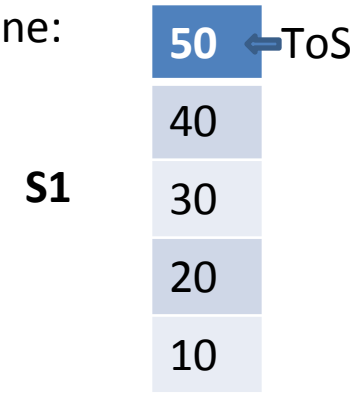
Queue using Stack

- Mechanism
  - We require 2 stacks to implement a Queue
    - Element will be inserted and deleted by following LIFO
    - Element need to be processed like FIFO concept
    - We require one Stack for data storing
    - We require another auxiliary Stack through which the data will be processed
    - We can accomplish the job by making data insertion phase costly or we can make the data deletion phase costly
      - Here, we will insert data normally but delete in costly manner

Queue using Stack

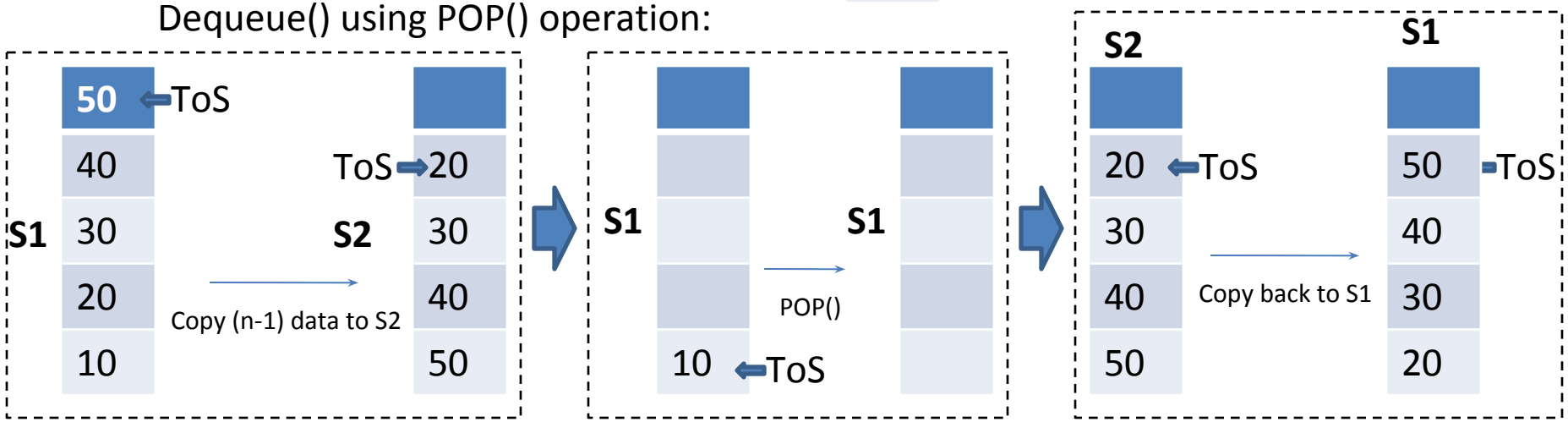
Data : 10 , 20 , 30 , 40 , 50

PUSH all data into stack1 one by one:



S1 : main stack
S2: auxiliary stack

Dequeue() using POP() operation:

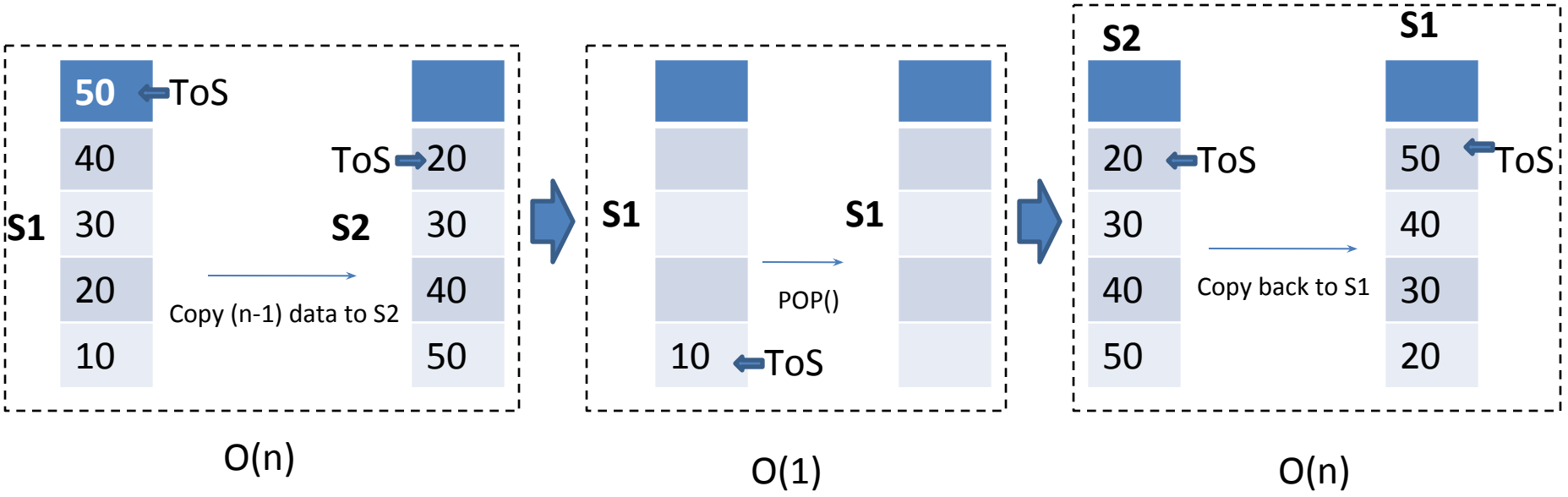


Queue using Stack

Time Complexity:

PUSH data into stack: O(1)

Dequeue() using POP() operation: O(n)



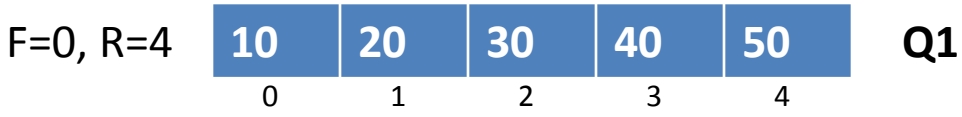
Stack using Queue

- Mechanism
  - We require 2 Queues to implement a Stack
    - Element will be inserted and deleted by following FIFO
    - Element need to be processed like LIFO concept
    - We require one Queue for data storing
    - We require another auxiliary Queue through which the data will be processed
    - We can accomplish the job by making data insertion phase costly or we can make the data deletion phase costly
      - Here, we will insert data normally but delete in costly manner

Stack using Queue

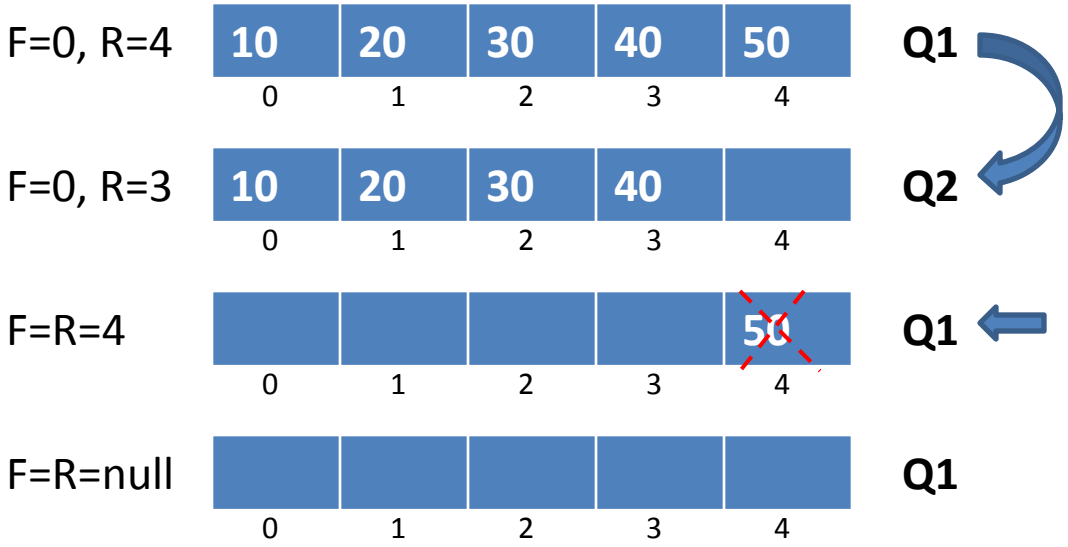
Data : 10 , 20 , 30 , 40 , 50

Enqueue all data into Q1 one by one:



Q1 : main queue
Q2: auxiliary queue

POP using Dequeue() operation:



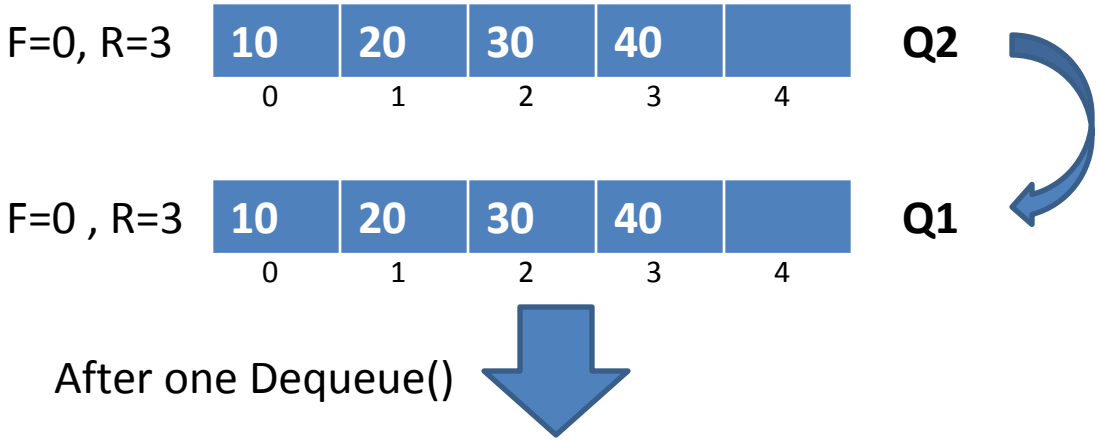
Contd…

Stack using Queue

Contd…

POP using Dequeue() operation:

Q1 : main queue
Q2: auxiliary queue



Copy all data from **Q2** back to **Q1**. Dequeue() from Q2 and Enqueue() in Q1

Last inserted data **50** is deleted from **Q1** i.e. it works as a Stack

Queries?