Fibonacci Series

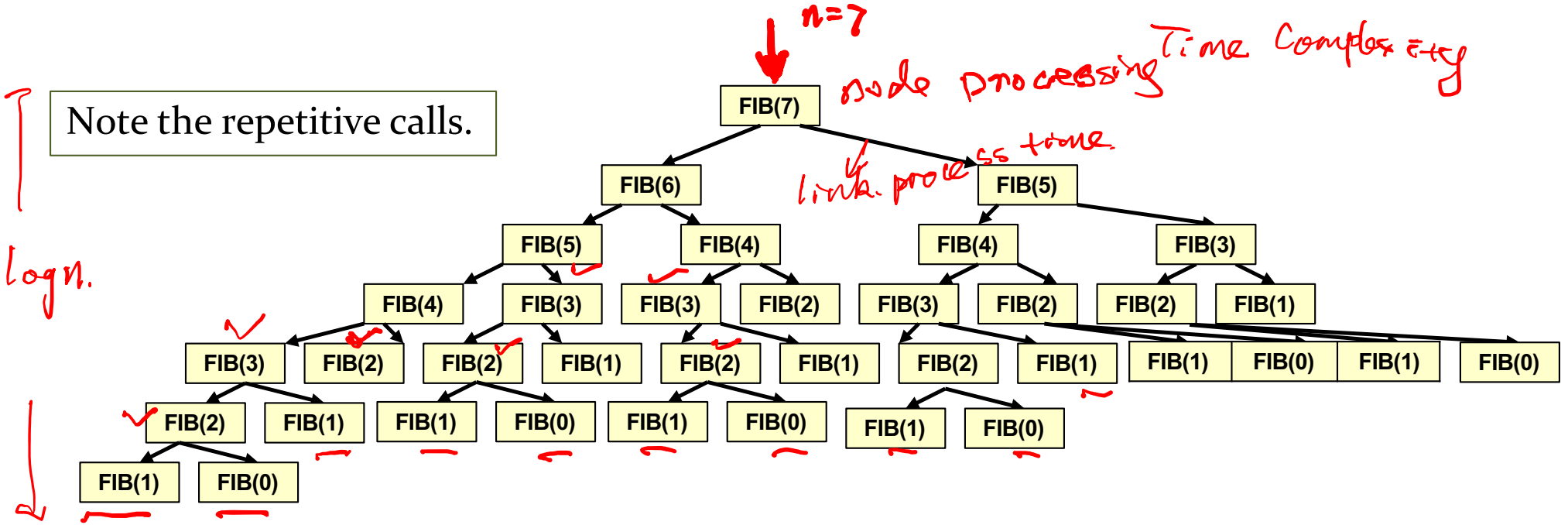
- The Fibonacci series can be given as:

0 1 1 2 3 5 8 13 21 34 55

.....

Fibonacci(n) = Fibonacci(n – 1) + Fibonacci(n – 2)

- Each term in the series is the sum of the two previous terms except for the first and second terms of 0 and 1.
- The recursive call FIB(7) has the following recursive call tree.



Pros and Cons of Recursion

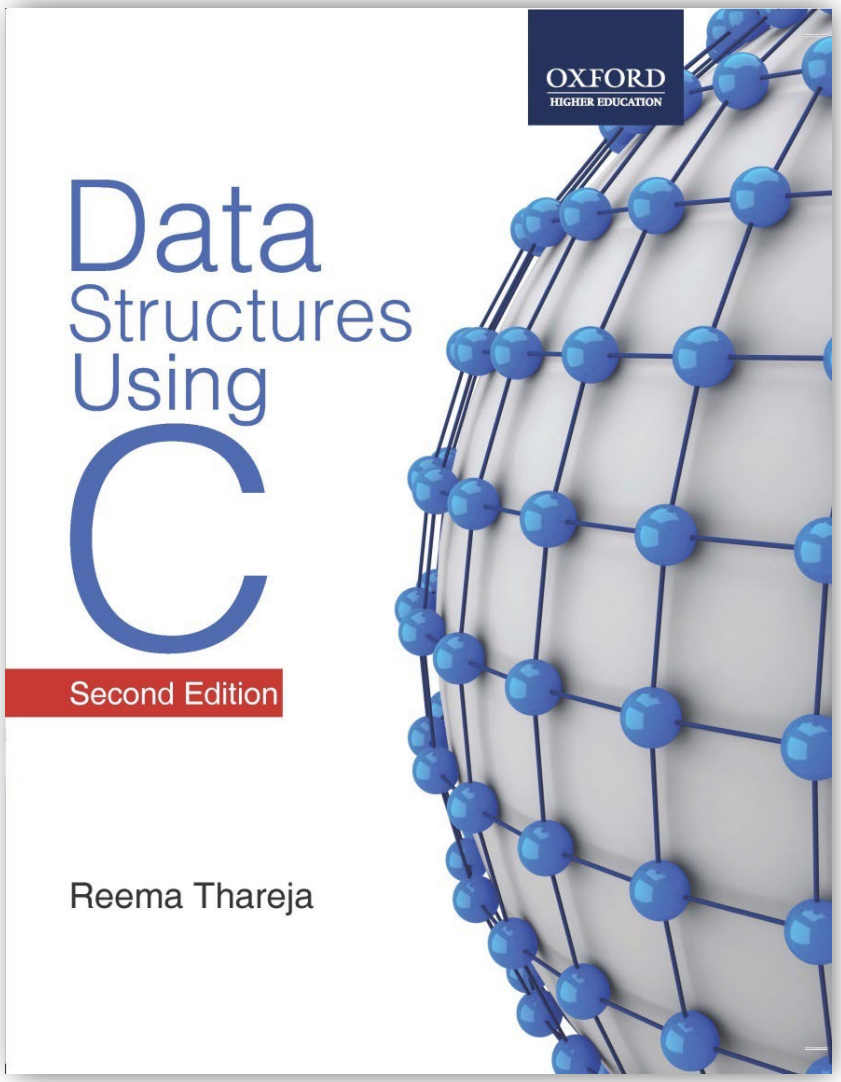
Pros

- Recursive solutions often tend to be shorter and simpler than non-recursive ones.
- Code is clearer and easier to use.
- Recursion follows a divide and conquer technique to solve problems.
- In some (limited) instances, recursion may be more efficient.

Cons

- Recursion is implemented using system stack. If the stack space on the system is limited, recursion to a deeper level will be difficult to implement.
- Aborting a recursive process in midstream is slow.
- Using a recursive function takes more memory and time to execute as compared to its non-recursive counterpart.
- It is difficult to find bugs, particularly when using global variables.

Thank you!



Data Structures Using C, 2e

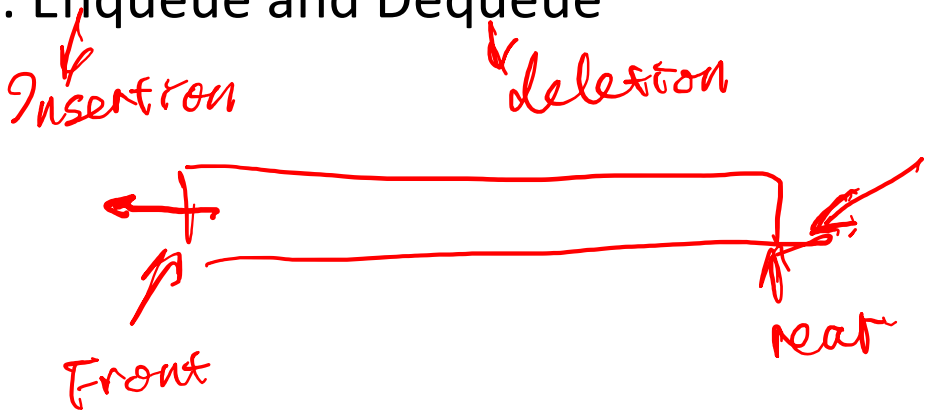
Reema Thareja

Chapter 8

Queues

Introduction

- A queue is a FIFO (First-In, First-Out) data structure in which the element that is inserted first is the first one to be taken out.
- The elements in a queue are added at one end called the *rear* and removed from the other end called the *front*.
- Queues can be implemented using arrays or linked lists.
- Basic operations: Enqueue and Dequeue



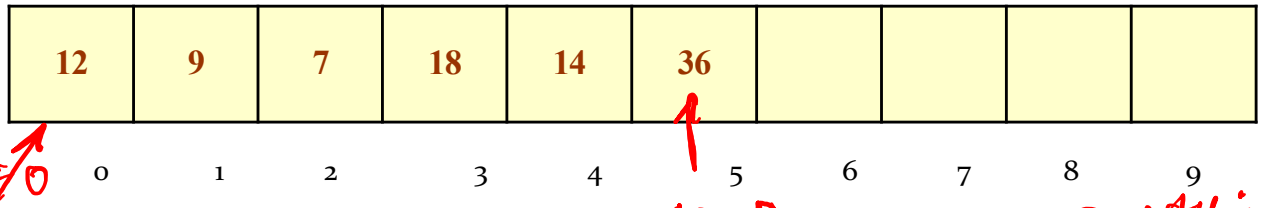
Applications of Queues

- Queues are widely used as waiting lists for a single shared resource like printer, disk, CPU.
- Queues are used to transfer data asynchronously, e.g. pipes, file I/O, sockets.
- Queues are used in Playlist for youtube to add songs to the end, play from the front of the list.

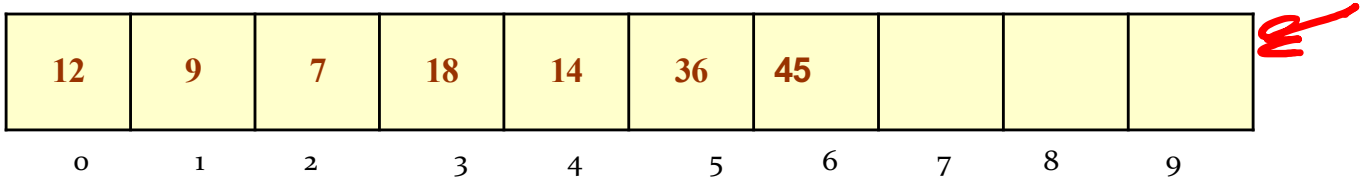
structure.

Array Representation of Queues

- Queues can be easily represented using linear arrays.
- Every queue has front and rear variables that point to the position from where deletions and insertions can be done, respectively.
- Consider the queue shown in the figure below.

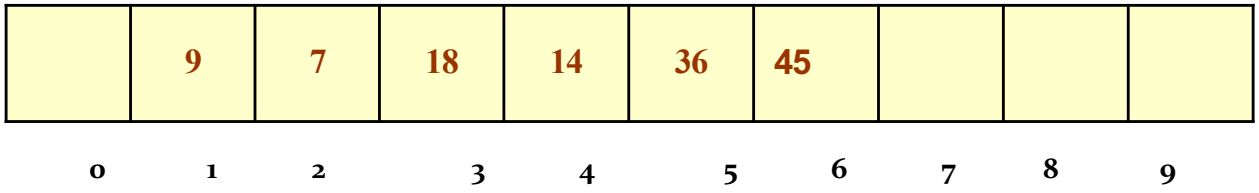


- Here, front = 0 and rear = 5.
- If we want to add one more value in the list say with value 45, then rear would be incremented by 1 and the value would be stored at the position pointed by rear.



Array Representation of Queues

- Now, front = 0 and rear = 6.
- Now, if we want to delete an element from the queue, then the value of front will be incremented.



- Now, front = 1 and rear = 6.

Array Representation of Queues

- Before inserting an element in the queue, we must check for overflow conditions.
- An overflow occurs when we try to insert an element into a queue that is already full, i.e. when rear = MAX – 1, where MAX specifies the maximum number of elements that the queue can hold.
- Similarly, before deleting an element from the queue, we must check for underflow condition.
- An underflow occurs when we try to delete an element from a queue that is already empty. If front = -1 or front > rear, this means there is no element in the queue.

Algorithm for Insertion Operation

```
Algorithm to insert an element in a queue
Step 1: IF REAR=MAX-1, THEN
            Write OVERFLOW
            Goto Step 4
        [END OF IF]
Step 2: IF FRONT == -1 and REAR = -1, THEN
          SET FRONT = REAR = 0
        ELSE
            SET REAR = REAR + 1
        [END OF IF]
Step 3: SET QUEUE[REAR] = NUM
Step 4: EXIT
```

Algorithm for Deletion Operation

```
Algorithm to delete an element from a queue
Step 1: IF FRONT = -1 OR FRONT > REAR, THEN
                  Write UNDERFLOW
        ELSE
                  SET VAL = QUEUE[FRONT]
                  SET FRONT = FRONT + 1
        [END OF IF]
Step 2: EXIT
```

Linked Representation of Queues

- In a linked queue, every element has two parts: one that stores data and the other that stores the address of the next element.
- The START pointer of the linked list is used as FRONT.
- We will also use another pointer called REAR which will store the address of the last element in the queue.
- All insertions will be done at the rear end and all the deletions will be done at the front end.
- If FRONT = NULL, then it indicates that the queue is empty.