Data Structures – Stack and Queue

ESC108A Elements of Computer Science and Engineering

B. Tech. 2017

Course Leaders:

Roopa G.

Ami Rai E.

Chaitra S.

Objectives

- At the end of this lecture, student will be able to
  - Explain the idea of a data structure
  - Use the structure and operations of a ***queue*** data structure
  - Use the structure and operations of a ***stack*** data structure

Contents

- Data Structure
- Arrays
- Queues
- Stacks

Effect of Data Organisation

- In a telephone directory
  - Find the phone number of any person by name of "Anil"
  - Find the owner's name for phone number 23333333

Data Structures

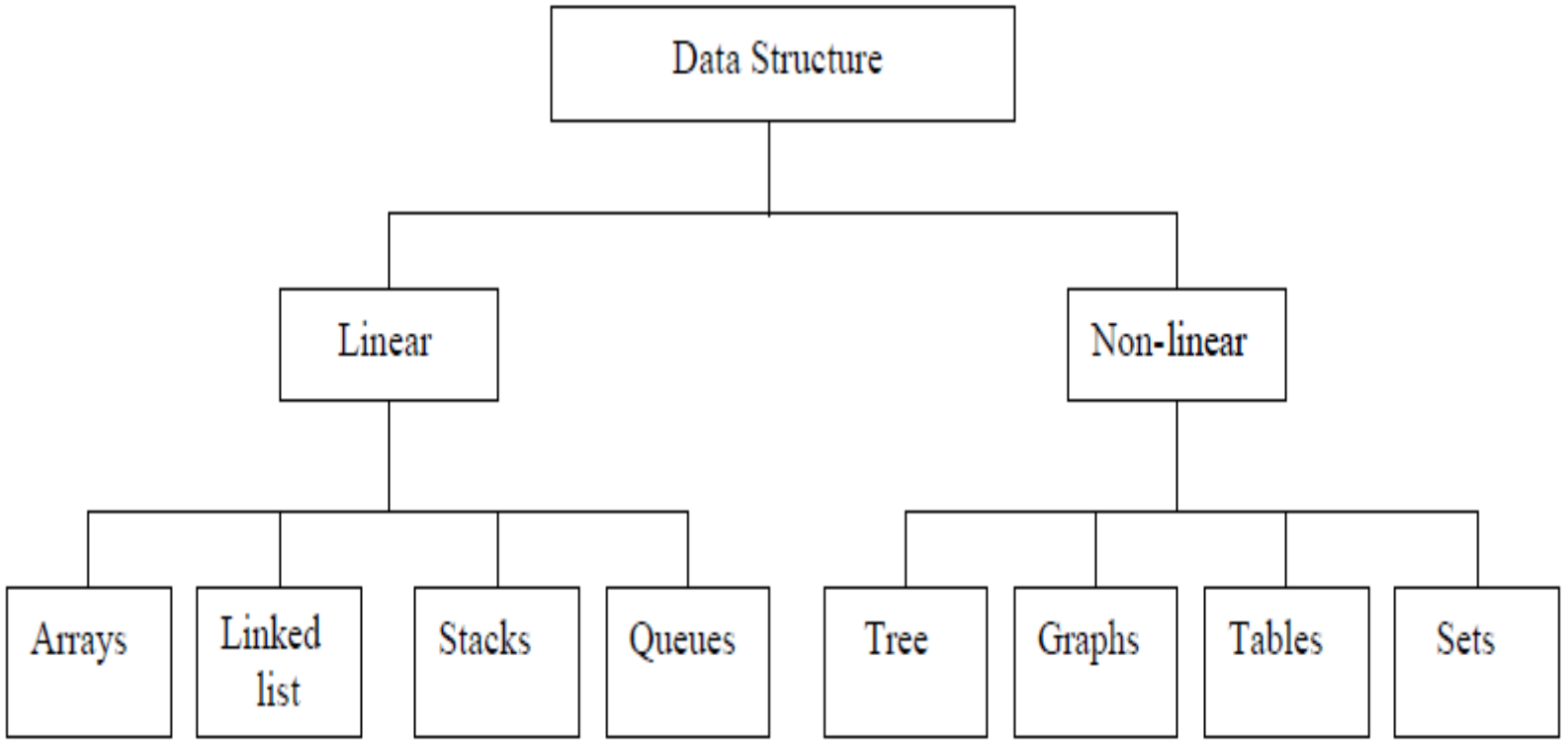
- A Data Structure is
  - A collection of elements of a type
  - Along with an set of operations defined on it

- Base of the programming tools
- Choice of data structure provides
  - The data structure should be satisfactory to represent the relationship between data elements
  - The data structure should be easy so that the programmer can easily process the data, as per requirement

Abstract Data Types (ADTs)

- Abstract Data Types (ADTs)
  - The Data Structure defined is independent of its implementation
  - This abstracted definition of a Data Structure and its Operations constitute the ADT
  - Provides a unified interface independent of the implementation details
  - May store redundant information to aid efficient operations to be performed on it

Classification of Data Structures

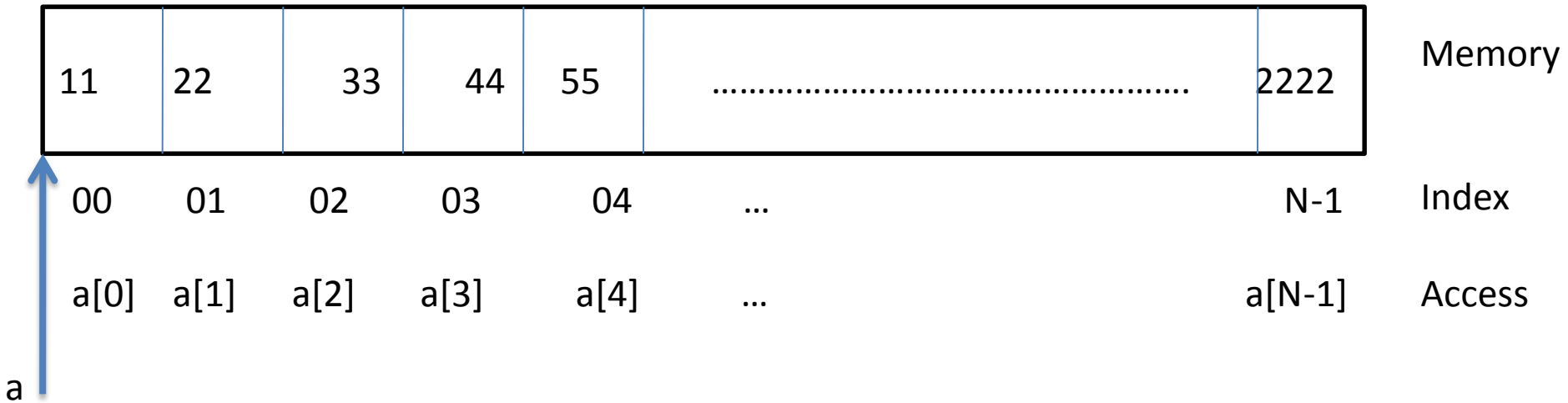


Data Structures: Arrays

- An array is the most basic data structure to visualise
- Most, if not all, computer languages provide it as a built-in
- Random access – accessing any element takes the same time
- Indexing could be either 0-based or 1-based

Arrays

- Arrays in C



- a is a constant and contains address of first element of the array

Application of Arrays

- Despite its simplicity, an Array has wide range of applications
- Best data structure for in memory storage of infrequently changing data
- Many sorting applications use arrays explicitly or as auxiliary data structures
- For implementation of other and more complicated data structures

  .. such as Stack, Queue, List, and Search Tree

Queue

- Represent waiting lines
  - the first person in line is serviced first, and other customers enter the line only at the end and wait to be serviced

- first-in, first-out (FIFO) data structure

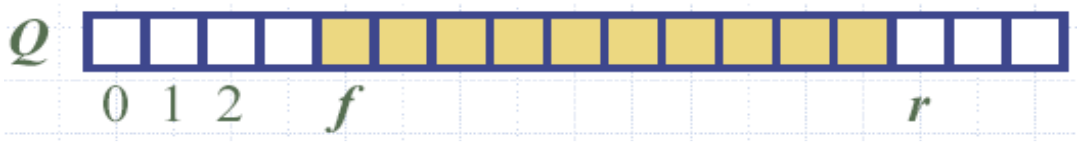
- Enqueue
  - Insertions are made at the back (tail of a queue)
- Dequeue
  - Deletions are made from the front (head of a queue)

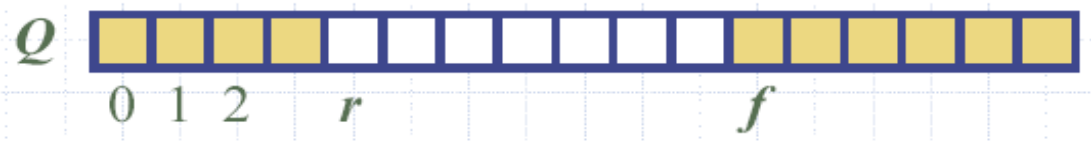
Queues in Computer Systems

- Accessing peripherals: Requests are queued
  - Print spooling - A multiuser environment may have only a single printer
- Kernel scheduling
  - CPU and other resources are scheduled by the kernel and made available to processes
  - Many computers have only a single processor, so only one user at a time may be serviced. Entries for the other users are placed in a queue
- Information packets wait in queues in computer networks
  - The routing node routes one packet at a time

Array Based Implementation of Queue

- Two approaches
  - Regular Array implementation
  - Circular Array implementation





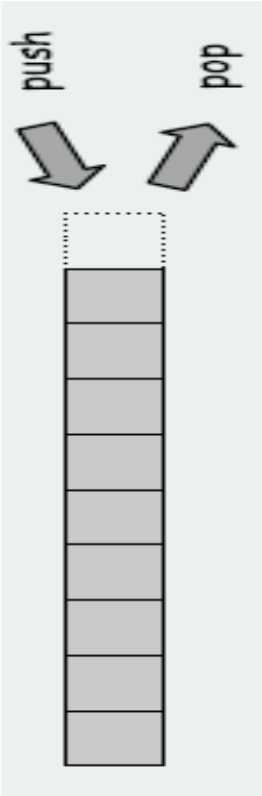
Array Based Realisation of Queue

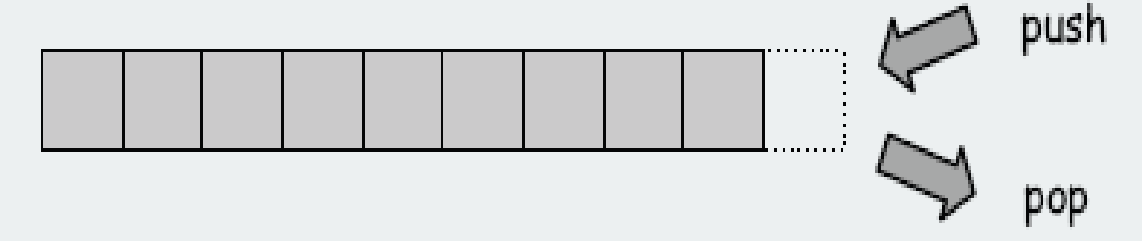
```
function size:
    return (N - f  + r) mod N
end function

function enqueue(e):
    if size = N - 1 then
        FullQueueException
    else
        Q[r] ← e
        r ← (r+1) mod N
    end if
end function
```

```
function isEmpty:
    return (f = r)
end function

function dequeue:
    if isEmpty then
        EmptyQueueException
    else
        o ← Q[f ]
        f  ← (f +1) mod N
        return o
    end if
end function
```

Stack

- Insertions and deletions are allowed at one end only ("top")
- Last-in,first-out (LIFO) data structure





Stack Operations

- The main Operations are
  - push(e): Puts element *e* at the top of the Stack
  - pop: Removes and returns the top-most element from the Stack

Overflow and Underflow

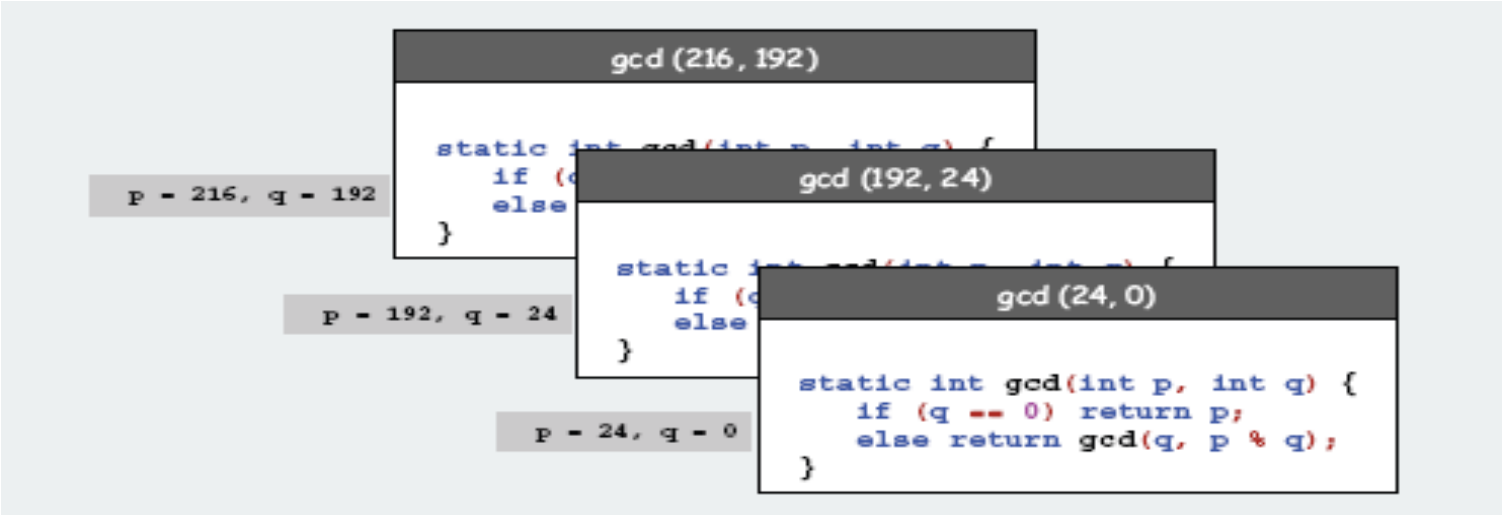
- It is important to consider the cases full Stack and empty Stack

- **Stack overflow**
  - When the stack contains equal number of elements as per its capacity and no more elements can be added

- **Stack underflow**
  - When there is no element in the stack or the stack holds elements less than its capacity

System Applications of Stack

- Stack is one of most used data structure for system programming
  - Compilers use it all the time to implement function calls
  - Function call: push local environment and return address
  - Return: pop return address and local environment
- Recursive function call:



  - One can always use an explicit stack to unravel a recursion

Function Call Stack

- When a program calls a function, the called function must know how to return to its caller, so the return address of the calling function is pushed onto the program execution stack
  - sometimes referred to as the function call stack

- Activation record or stack frame of the function call
  - a portion of the program execution stack
  - memory for the local variables used in each invocation of a function during a program's execution

Function Call Stack contd.

- When a function call is made
  - the activation record for that function call is pushed onto the program execution stack
- When the function returns to its caller
  - the activation record for this function call is popped off the stack and those local variables are no longer known to the program

- stack overflow
  - Error occurs if more function calls occur than can have their activation records stored on the program execution stack

Implementation of Stack

- The stack implementation can be done in the following ways:
  1. Using arrays
  2. Using linked list

Array Based Stack Implementation

```
function size:
  return 1+t
end function

function push(e):
    if t = N - 1 then
        StackFullException
    else
        t ← t + 1
        S[t] = e
    end if
end function
```

```
function pop:
    if isEmpty() then
        StackEmptyException
    else
        o ← S[t]
        t ← t – 1
        return o
    end if
end function
```

Summary

- ADT is an abstract model of a data structure along with its operations
- ADT makes it possible to express algorithms independent of the underlying implementation of the data structure
- Array and List data structures are fundamental building blocks of other data structures
- Stack and Queue ADTs are data structures in which there is restriction on the insertion and deletion of elements
- In a Stack, elements can be inserted and removed only at one end
- In a Queue, elements can be inserted at one end and can be removed only from the other end