
# Introduction to Computing

Lectured by: Dr. Pham Tran Vu

t.v.pham@cse.hcmut.edu.vn


# Programming

- Programming languages
- Program design, testing, debugging and documenting
- **Data structures**


# Content

- Arrays
- Stack
- Queue
- Circular queue
- Linked list
- Binary tree


# Data Structures

- Essential for any computer programmer
- A data structure is essentially a number of data items, with some relationships among them
- Each item occupies one or more memory location


# Arrays

- Supported by all high-level programming languages, such as Pascal, C/C++, Java, etc
- Data is manipulated in tabular fashion
- Programmers only need to declare the array name, size and dimension, the language processor will allocate memory for the array
- An array can be declared as one-dimension, two dimension, and so on

# One-Dimension Arrays

- Allocating one-dimension array in memory

Memory

| Address | Content |
|---|---|
| … | |
| b | |
| b+1 | |
| b+2 | |
| b+3 | |
| … | |

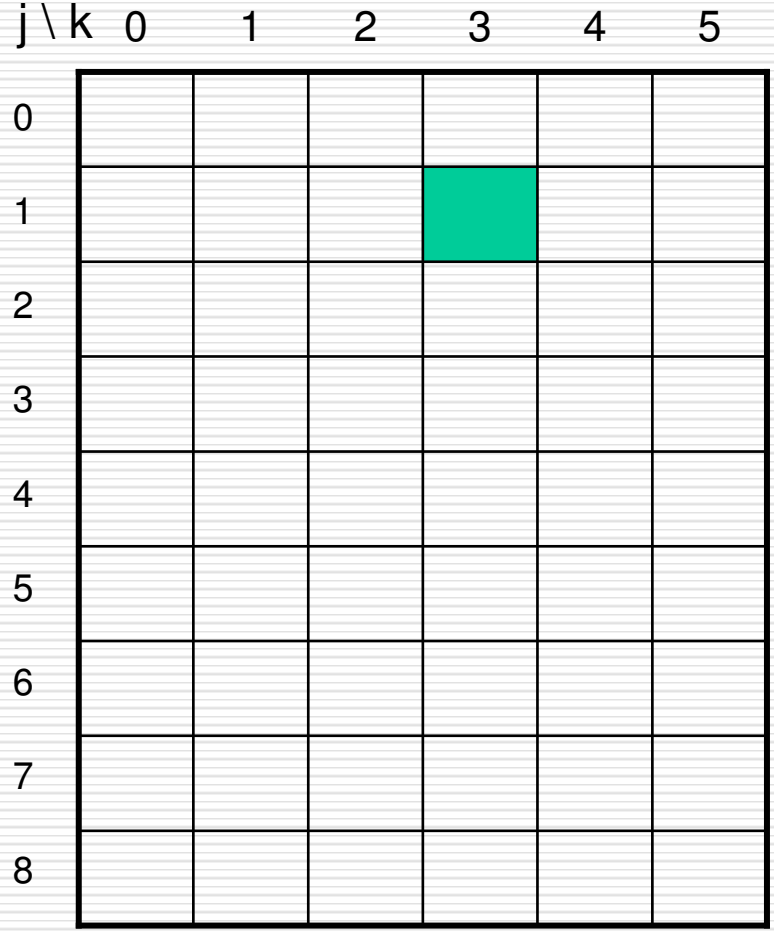
Array

| Subscript | Content |
|---|---|
| 0 | |
| 1 | |
| 2 | |
| 3 | |
| … | |


# Two-Dimension Arrays

- Memory addressing is one-dimension
- Need a translate of subscripting to mapping two-dimension array addressing into memory address

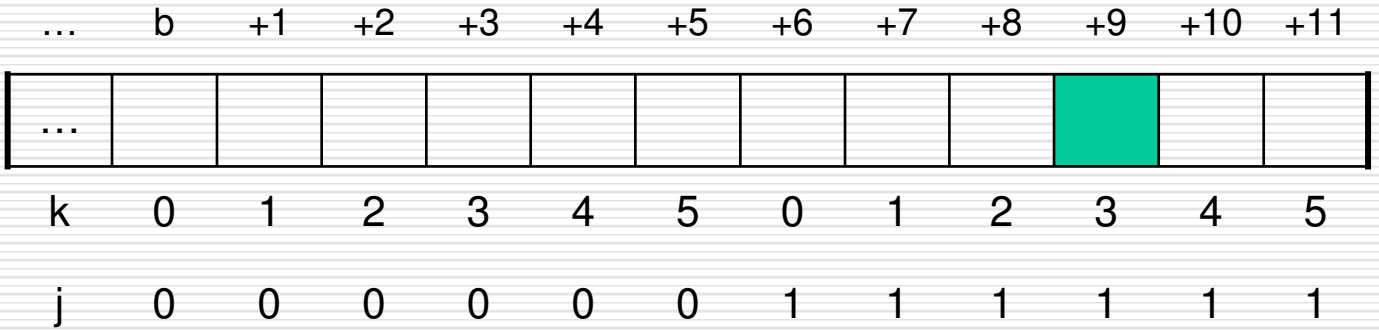
# Two-Dimension Arrays (2)




A[1][3]
j = 1
k = 3

# Two-Dimension Arrays

- Storage of two-dimension array in memory

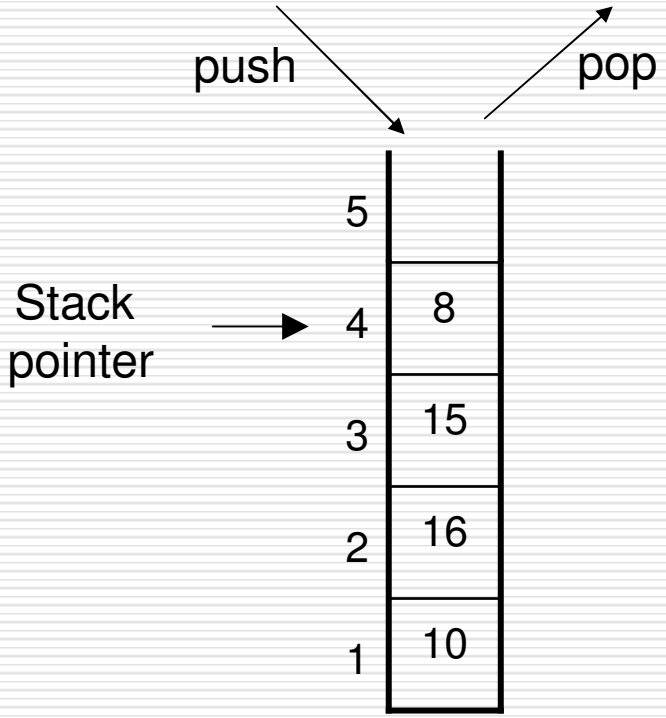



- Translation formula: M = b + j*N + k
  - N: array size
  - M: memory location of element A[j][k]
  - b: starting memory address of the array


# Stacks

- Characterised by Last In First Out (LIFO)
- Items are added or removed from a stack is done at only one end
- Two basic operations:
  - Push: store an item into the stack
  - Pop: get an item out of the stack

# Stacks (2)




- A stack of size 5
- Currently store 4 item
- Current stack pointer value is 4
- A stack can be implemented using a one-dimension array

# Stack push operation

```
If sp < maximum stack size
   Then increase sp by 1
      S(sp) = item
   Else stack overflow
End if
```

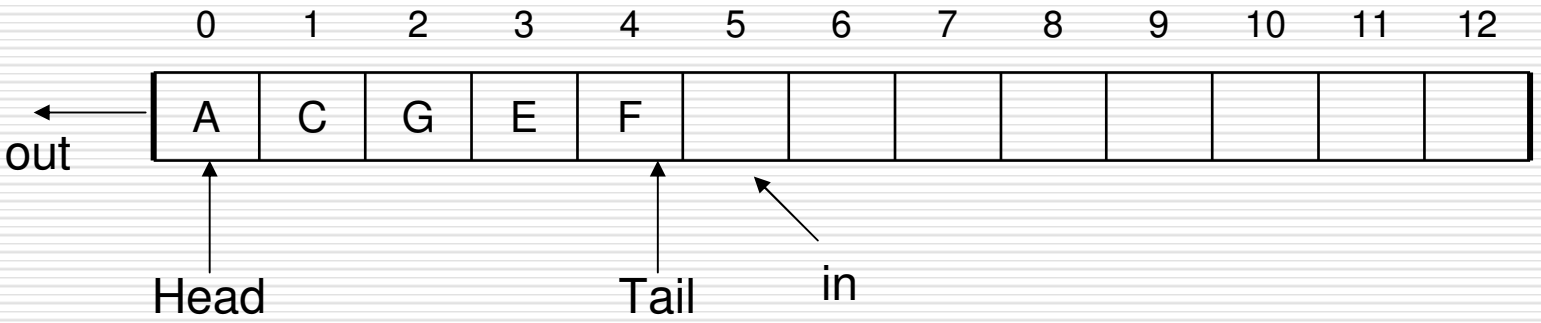
- S: the stack
- sp: the stack pointer

# Stack pop operation

```
If sp > 0
   Then item = S(sp)
   Sp = sp – 1
   Else stack underflow
End if
```

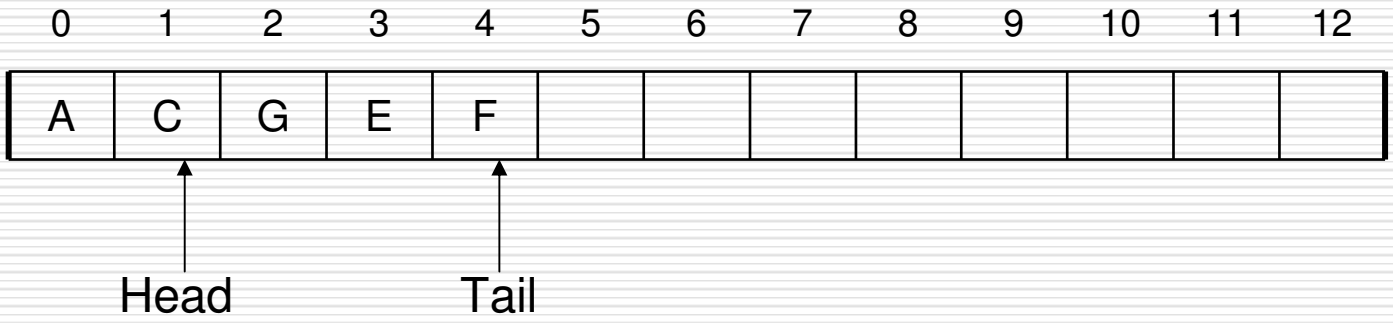
# Queues

- Characterised by First In First Out FIFO
- Similar to our every queues
- In a queue, data item is added to one end, and is remove at the other end of the queue



# Queue Implementation Using One-Dimension Array

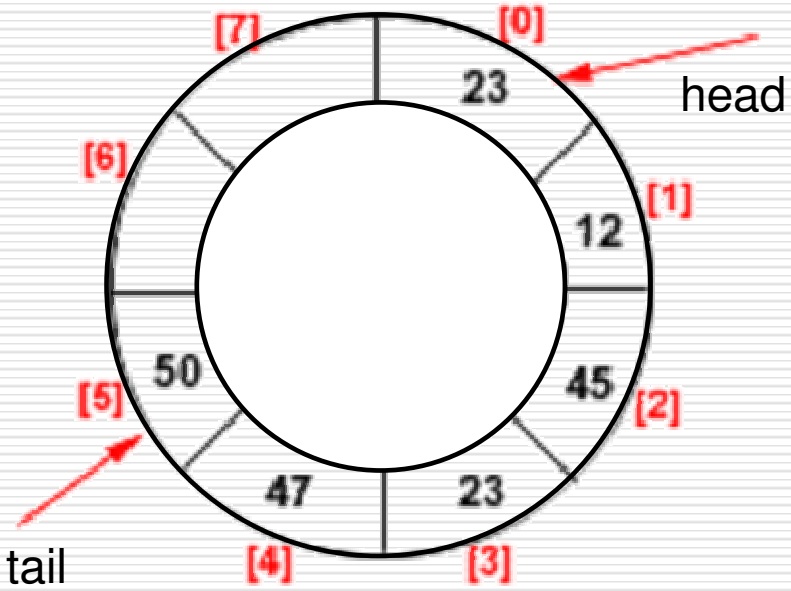
- Implementation of a queue using array of 13 elements
- To add an item, the Tail is increased by 1
- To remove an item the Head is increased by 1
- The queue is full when Tail equals 12
- The queue is empty when Head equals Tail



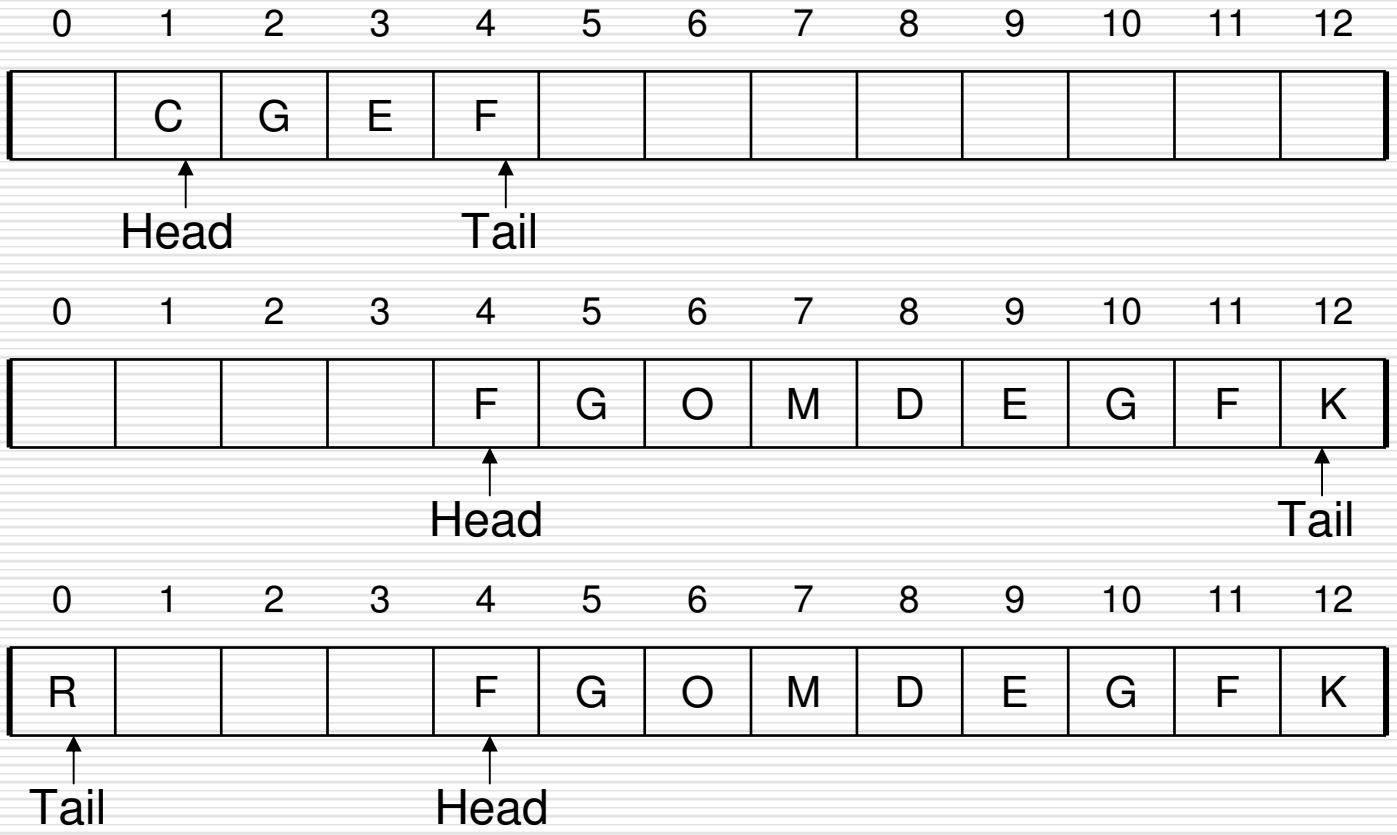

# Circular Queues

- The disadvantage of the implementation discussed is that the queue will be definitely full after some usage, although the number of items in the queue is still smaller than the size of the queue
- A circular buffer implementation can be used to solve this problem

# Circular Queues (2)



# Array Implementation of Circular Buffer



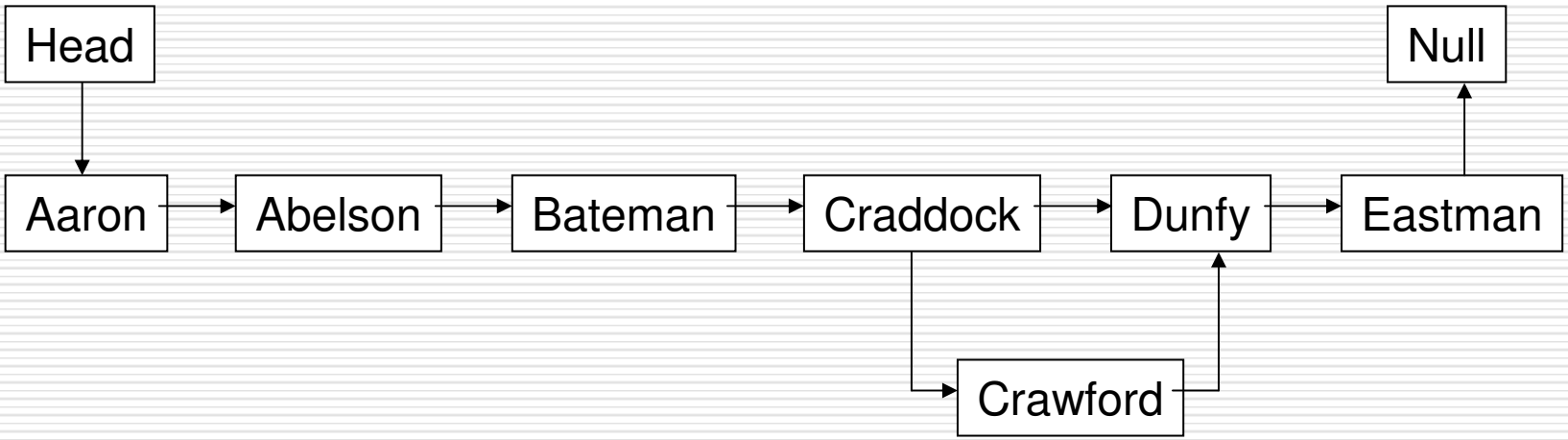

# Linked Lists

- Give the array contains order list of names
- How to add in Crawford, while the alphabetical order of the list is still maintained?

| Element | Data |
|---|---|
| 0 | Aaron |
| 1 | Abelson |
| 2 | Bateman |
| 3 | Craddock |
| 4 | Dunfy |
| 5 | Eastman |
| 6 | |
| 7 | |

# Linked Lists (2)

- In a linked list, an element has a pointer to the element follows it

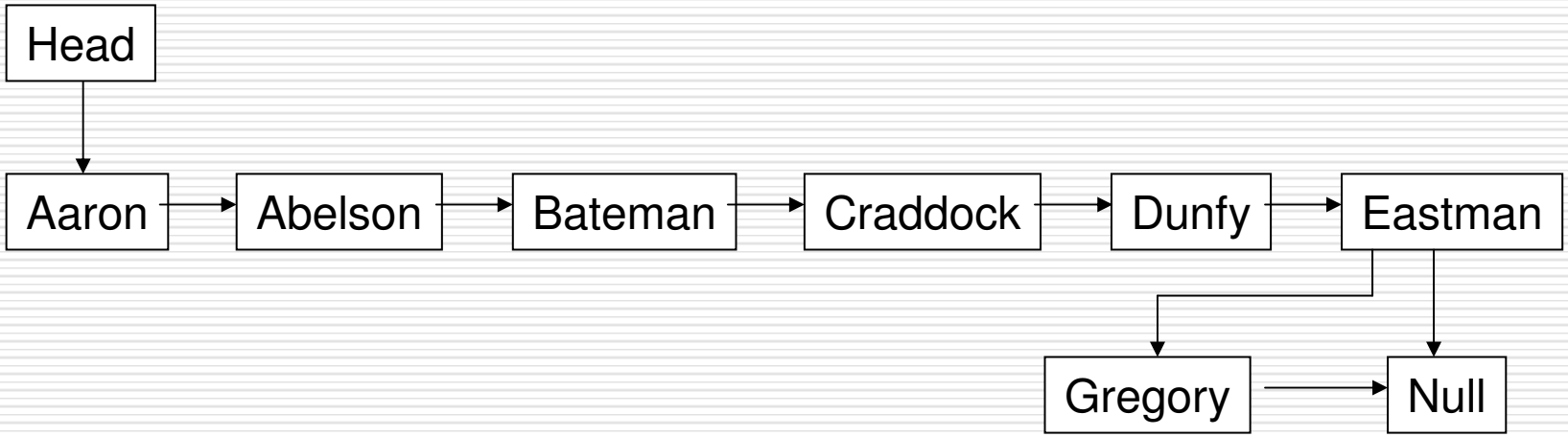



# Linked List Operations

- Add an element
- Delete an element
- Insert an element

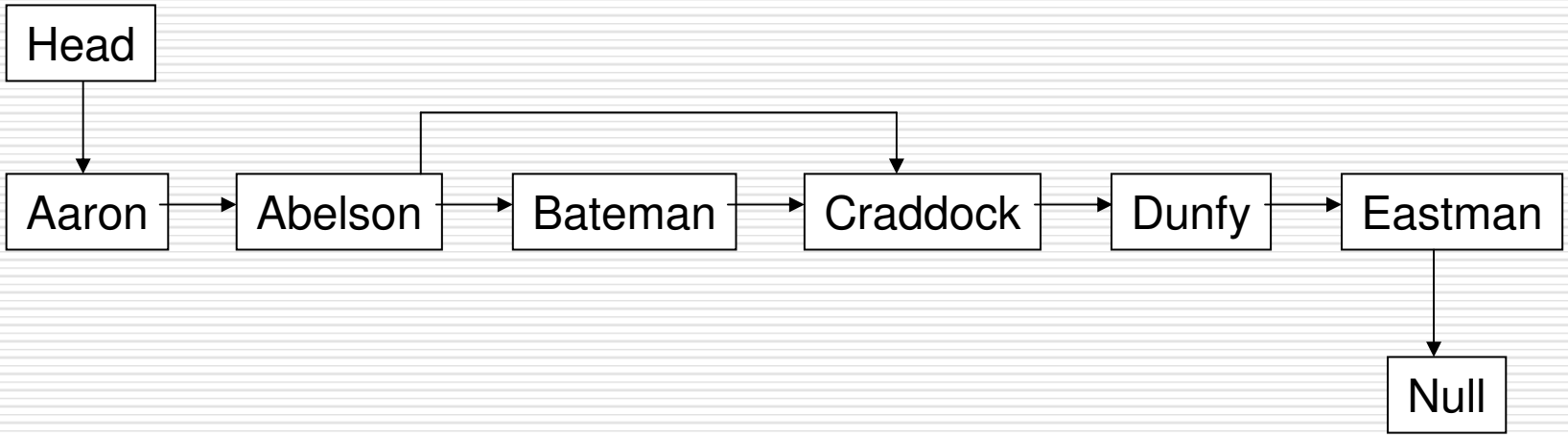
# Add an Element to the End of the List

- Add Gregory to the end of the list



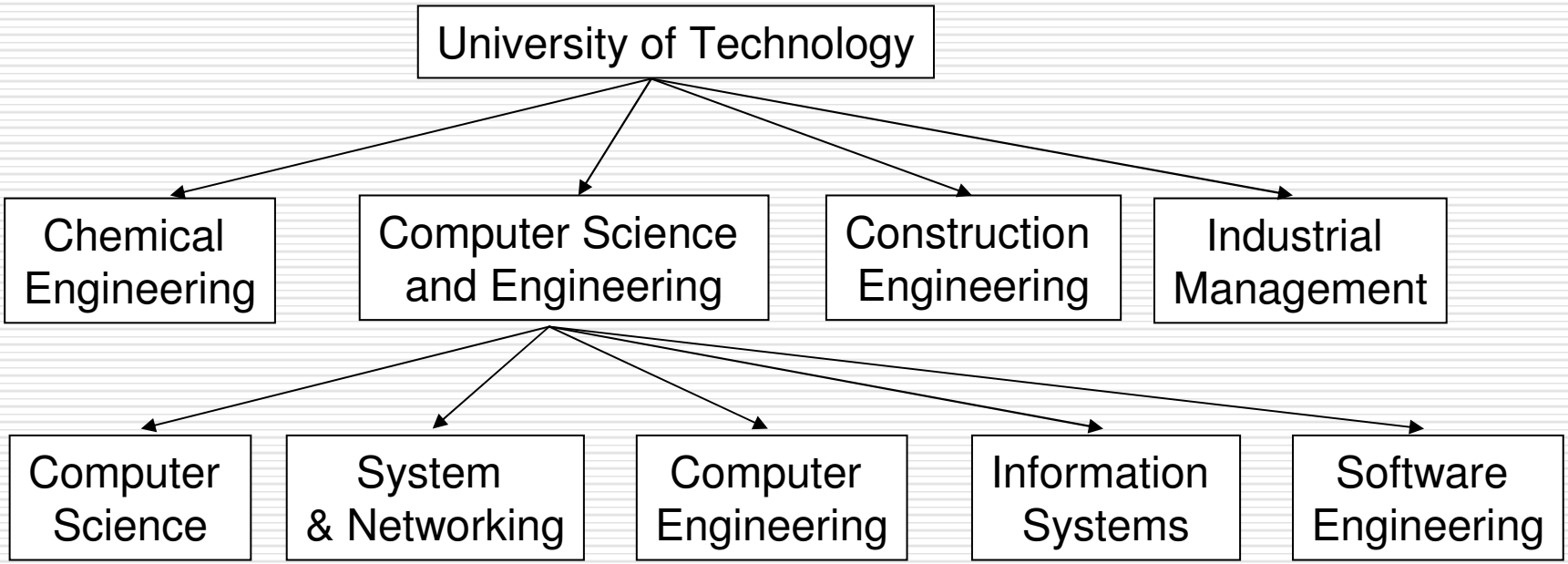
# Delete an Element

- Delete Bateman from the list



# Trees

- In a tree data structure, data elements are organised in a tree like structure

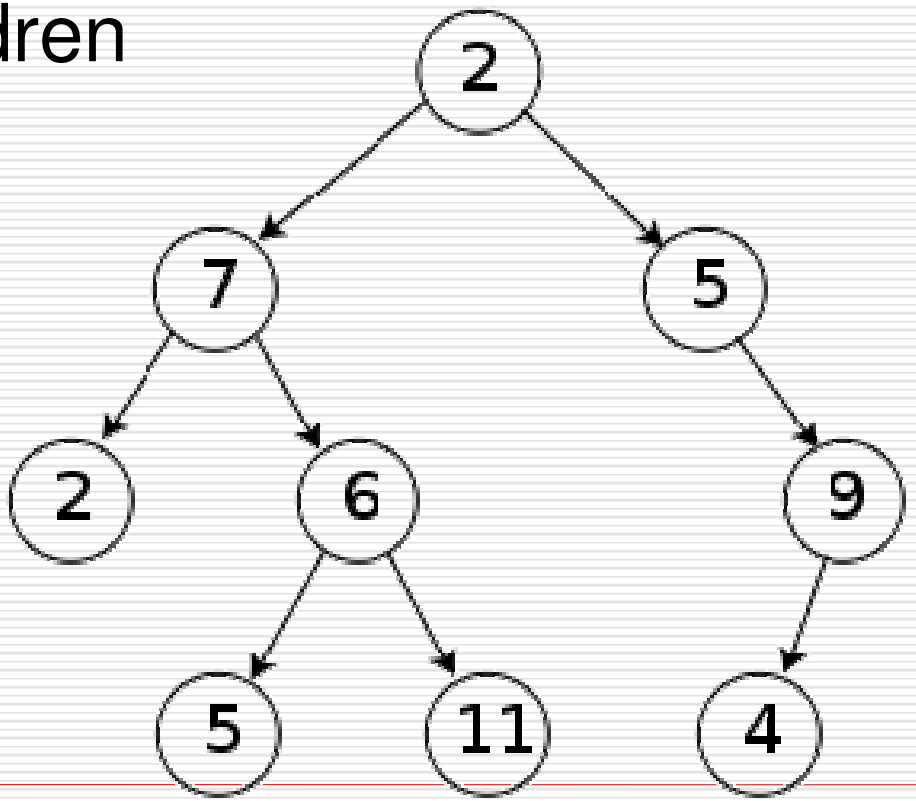



# Tree Data Structure Concepts

- A parent node: the node that has pointer(s) to other node(s)
- A child node: the one that is pointed to a parent node
- Root: the node of the tree that has no parent
- Leaf node: the node that has no child
- A node in a tree can’t have more than one parent

# Binary Tree

- In a binary tree, each node can has at most two children




# Binary Search Trees

- A binary search tree is a binary tree that has the following characteristics
  - Each node has a distinct value
  - Values of all the nodes of the left subtree of a node are smaller than the node's value
  - Values of all the nodes of the right subtree of a node are greater than the node's value

# Binary Search Trees (2)