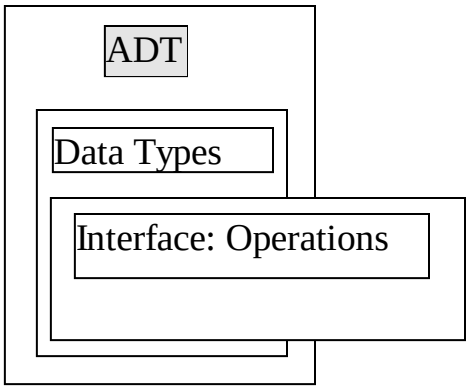
Course Summary

- **Performance Analysis**
  - Estimation of required resources such as memory space, computational time, and communication bandwidth.
  - Comparison of algorithms
  - Common time functions:

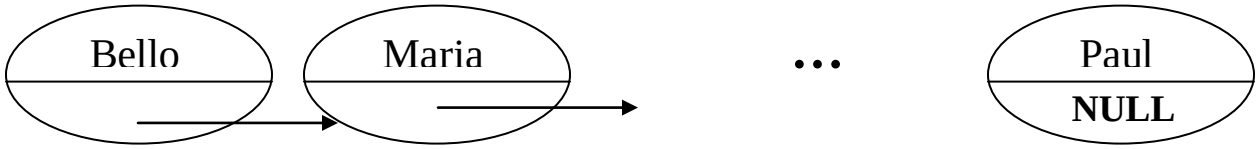
If F is:	Say that F is:	If F is:	Say that F is:
O(1)	Constant	$O(n^r)$, $1<r<2$	Subquadratic
O(logn)	Logarithmic	$O(n^2)$	Quadratic
$O(\log^c n)$, $c>=1$	Polylogarithmic	$O(n^3)$	Cubic
$O(n^r)$, $0<r<1$	Sublinear	$O(n^c)$, $c>=1$	Polynomial
O(n)	Linear	$O(r^n)$, $r>1$	Exponential

- **Abstract Data Type: ADT**
    - An abstract data type (ADT) is characterized by the following properties:
        - It exports a type, called domain.
        - It exports a set of operations. This set is called **interface**.
        - Operations of the interface are the one and only access mechanism to the type's data structure.
        - Axioms and preconditions define the application domain of the type.



- **Arrays:**
  - Arrays are one the main data organization
  - Arrays are very efficient way of organizing data since accessing array elements requires O(1).
  - **Two-dimensional arrays: Matrices**
    - Two-dimensional arrays are also called table.
    - Requires two dimensions:
      - Rows
      - Columns
    - Internal Representation:
      - How is a 2-dimensional array stored in the sequential memory?
      - Two common schemes:
        - **Row major order**: rows are placed one after another in memory. Examples: Java, C, C++, Pascal, etc.
        - **Column major order**: columns are placed one after another in memory. Example: Fortran.

- **Linked Lists:**
  - A linked list consists of a number of nodes, each of which has a reference to the next link.

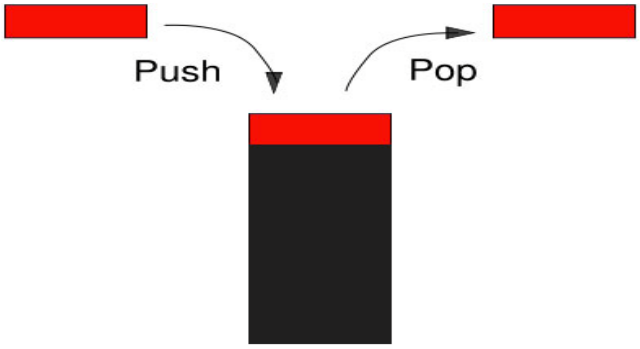


  - **Think about the efficiency of each operation:**
    - Insert
    - Delete
    - Search

  - **Singly Linked Lists:**
    - One-way linked lists
    - One-way linked lists with head and tail
    - Circular one-way linked lists

  - **Doubly Linked Lists:**
    - Simple doubly linked lists
    - Circular doubly linked lists

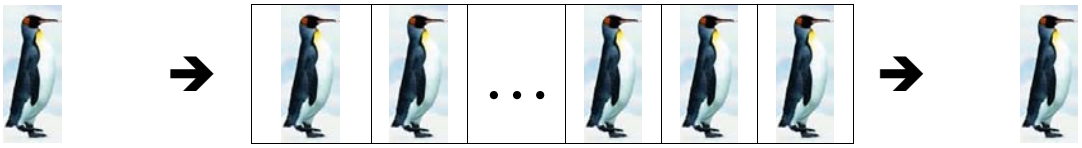
- **Stack:**
  - It is an LIFO ADT
  - It is a list of elements where a new element is inserted or pushed and deleted or popped only at the top of the list.
  - Only the element at the top of the list is accessed
  - Example:



  - Efficiency:

Big-O Comparison of Stack Operations		
Operation	Array Implementation	Singly Linked Implementation
Class constructor	O(1)	O(1)
Empty The Stack	O(1)	O(N)
IsFull?	O(1)	O(1)
IsEmpty?	O(1)	O(1)
Push	O(1)	O(1)
Pop	O(1)	O(1)
Destructor	O(1)	O(N)

- **Queue:**
  - It is an FIFO ADT
  - A new element is added or inserted to the end of the list
  - An element is deleted or removed only from the beginning of the list.



Enqueue Dequeue

  - **Efficiency:**

Big-O Comparison of Queue Operations			
Operation	Array Implementation	Singly Linked Implementation	Linked List with Head and Tail Implementation
Class constructor	O(1)	O(1)	O(1)
MakeEmpty	O(1)	In Java, O(1) Others: O(N)	In Java, O(1) Others: O(N)
IsFull	O(1)	O(1)	O(1)
IsEmpty	O(1)	O(1)	O(1)
Enqueue	O(1)	O(1)	O(1)
Dequeue	O(1)	O(N)	O(1)
Destructor	O(1)	O(1)	O(1)

- **Recursion:**
  - A procedure or function that calls itself, directly or indirectly, is said to be recursive.
  - Format of a recursive algorithm:

    - Algorithm name(parameters)
      Declarations;
      Begin
        if trivial case)
        then do trivial operations
        else begin
          - one or more call name(smaller values of parameters)
          - do few more operations: process the sub-solution(s).
        end;
      end;
  - **Performance**
    - It uses a recurrence relation
    - A recurrence relation of a sequence of values is defined as follows:
      - (B) Some finite set of values, usually the first one or first few, are specified.
      - (R) The remaining values of the sequence are defined in terms of previous values of the sequence.

- Example:

```
procedure Quicksort(type A[1..n], int p,r);
        int q;
        Begin
                if (p<q)                                    ➔ O(1)
                then begin
                                q = partition(A[1..n], p,r);    ➔ O(n)
                                Quicksort(A[1..n], p,q);        ➔ T(n/2)
                                Quicksort(A[1..n], q+1,r);      ➔ T(n/2)
                        end;
                endif;
        end;
```

- So the time complexity function (recurrence relation):

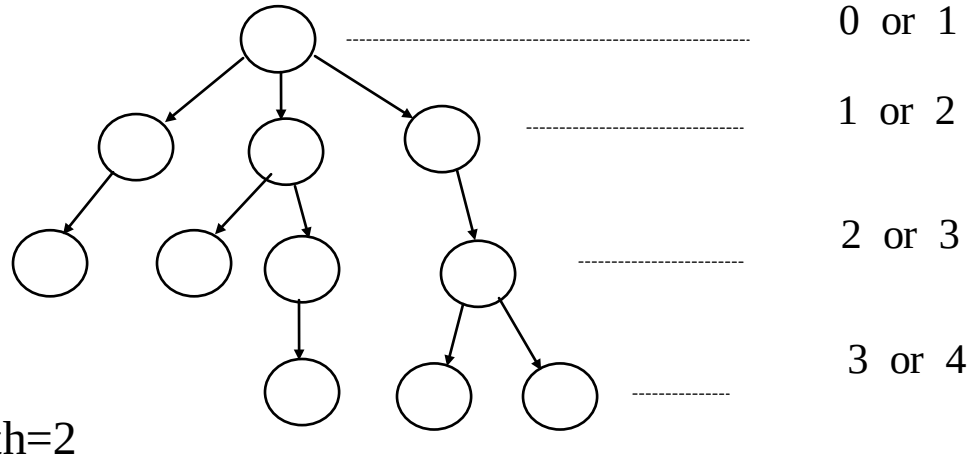
  $T(n) = 2T(N/2) + n$ if $n>1$

  $T(1) = 1$

- **Graphs**:
  - A graph G is an ordered pair of sets (V,E) where V is a set of nodes and E is a set of edges or (arcs).
  - There are two types:
    - directed graphs (Digraphs)
    - undirected graphs
  - Graph Representation
    - There are two types of representations:
      - Adjacency matrix
      - Adjacency list
  - Efficiency:
    - Algorithms using adjacency matrix representation require at least $O(n^2)$ where n=|V| and V is the set of vertices of the input graph.
    - Algorithms using adjacency list representation require at least O(max(n,e)) where n=|V| and V is the set of vertices of the input graph and e=|E| and E is the set of vertices.

  - Graph Traversals
    - There are two strategies:
      - **Depth First Search (DFS)**
      - **Breadth First Search (BFS)**

- Trees:
  - A **tree** is a connected acyclic graph.
  - A disconnected acyclic graph is called a **forest**
  - A tree is a connected digraph with these properties:
    - There is exactly one node (**Root**) with in-degree=0
    - All other nodes have in-degree=1
    - A **leaf** is a node with out-degree=0
    - There is **exactly one path** from the root to any leaf
  - The **degree** of a tree is the maximum out-degree of the nodes in the tree.
  - Level of a node:

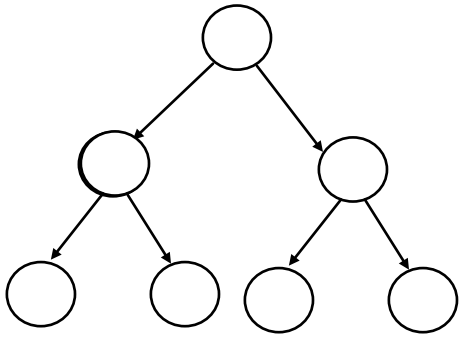


  - **Properties**:

    (1) for a tree T =(V,E), where n=|*V*| and e=|*E*|, we have

$$e = n - 1$$

  - **Binary Trees**:
    - It is a tree whose **degree is ≤ 2**
    - The two children are called **left and right** children



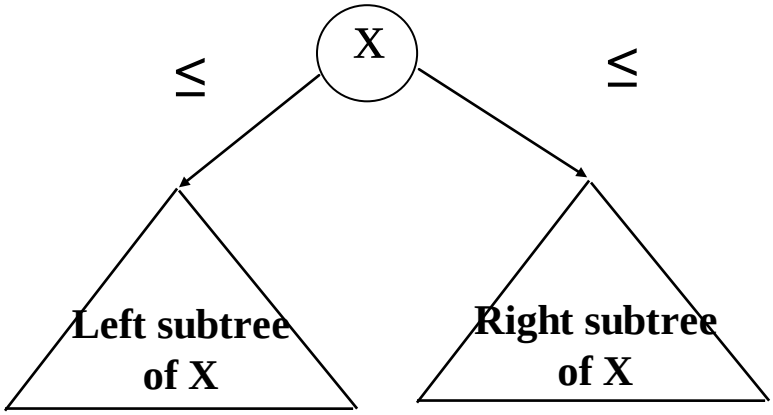
- **Lemmas**:
  - The maximum number of nodes on level i of a binary tree is 2^i (starting from level 0).
  - The maximum number of nodes in a binary tree of depth k is: 2^{k+1}-1, k>0 (starting from level 0).d
  - For any non empty binary tree, T, if n_0 is the number of leaves and n_2 is the number of nodes of degree 2, then

$$n_0 = n_2 + 1$$

- Representations:
  - Sequential
  - Linked-list

- Binary Tree Traversals:
  - Inorder: LNR
  - Preorder: NLR
  - Postorder: LRN

- Binary Search Trees:
  - Insertion, deletion, and Find take O(log(n)) where n is the number of elements in the tree.
  - BST property:

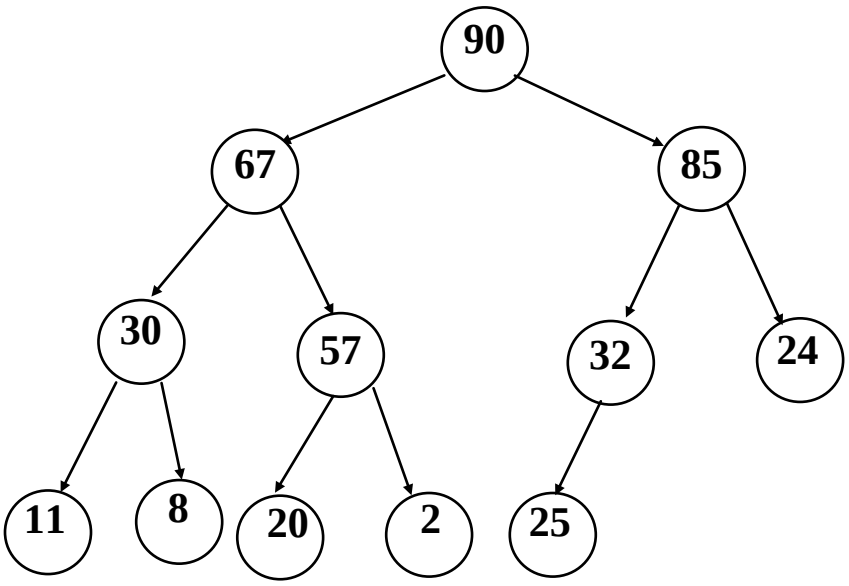


  - Operations:
    - Search or Find
    - Insert
    - Delete
    - Find_min
    - Find_max

- **Priority Queue:**
  - A priority queue is a restricted form of a list, where items are arranged according to their priorities (or keys). The key assigned to an item may not be unique.
  - The item with highest priority is removed in O(1).
  - Each node stores prioritized key-item(s) pairs

  - Heap
    - Heap is a priority queue.
    - Get an object with highest priority in a constant of time O(1).
    - Heap Structures
      - A **max-heap** (**min-heap**) is a complete BT with the property that the key (priority) of each node is at least as **large** (**small**) as the values at its children (if they exist).

    - Implementation:
      - Sequential representation
    - Example:



- **Operations**:
  - Insert
  - Delete
  - Delete max/min
  - Get max/min

- **Sorting**
  - An algorithm that segregates order items according to specified criterion: Ascending or descending
  - Recursive and non-recursive algorithms
  - Comparing Sorting Algorithms:

	Worst Case	Average Case
Selection Sort	$O(n^2)$	$O(n^2)$
Bubble Sort	$O(n^2)$	$O(n^2)$
Insertion Sort	$O(n^2)$	$O(n^2)$
Mergesort	O(nlogn)	O(nlogn)
Quicksort	$O(n^2)$	O(nlogn)
Heapsort	O(nlogn)	O(nlogn)