Analysis of Algorithms - CS 323/700
Lecture #2 - February 10, 2016

Notes by: Ammad Khan

Content of the lecture:

- Homework #1 review
- Characteristic equation to find a closed form formula for the Fibonacci sequence
- Concept of best, average and worst case time complexity
- Fundamental data structures
    - Stacks
    - Queues
    - List
    - Dictionary
    - Trees
    - Heap

1. **Derive a formula for the *sum of squares* $1^2 + 2^2 + 3^2 + \ldots + n^2$. Hint: assume the formula is a polynomial of degree 3, i.e. $an^3 + bn^2 + cn + d$, and use the cases of n=0, n=1, n=2, and n=3 to determine its coefficients.**

Solution:

- First we find the upper bound and the lower bound of the given sum of square equation. n^2 is the lower bound because whatever n is set to be the lowest term which will be a square term. n^3 is the upper bound as the square of all term cannot be larger than the term.

$$n^2 < 1^2 + 2^2 + 3^2 + \ldots + n^2 < n^3$$

- Therefore we can represent the sum of square by a polynomial of degree 3

$$1^2 + 2^2 + 3^2 + \ldots + n^2 = an^3 + bn^2 + cn + d$$

- In order to find the constants (a, b, c) of the polynomial we set the given equation with n=0, 1,2, 3

n=0 : $a(0)^3 + b(0)^2 + c(0) + d = 0$
$d=0$

n=1 : $a(1)^3 + b(1)^2 + c(1) + d = 1$
$a + b + c = 1$ ----------------- Equation 1

n=2 : $a(2)^3 + b(2)^2 + c(3) + d = 1 + 4$

$8a + 4b + 2c = 5$ ----------------- Equation 2

n=3 : $a(2)^3 + b(2)^2 + c(3) + d =$

$27a + 9b + 3c = 14$ ----------------- Equation 3

- Using equation 1,2 and 3 with the wolfram tool provided to solve the linear equation.

  $a = 1/3$ $b = 1/2$ $c = 1/6$

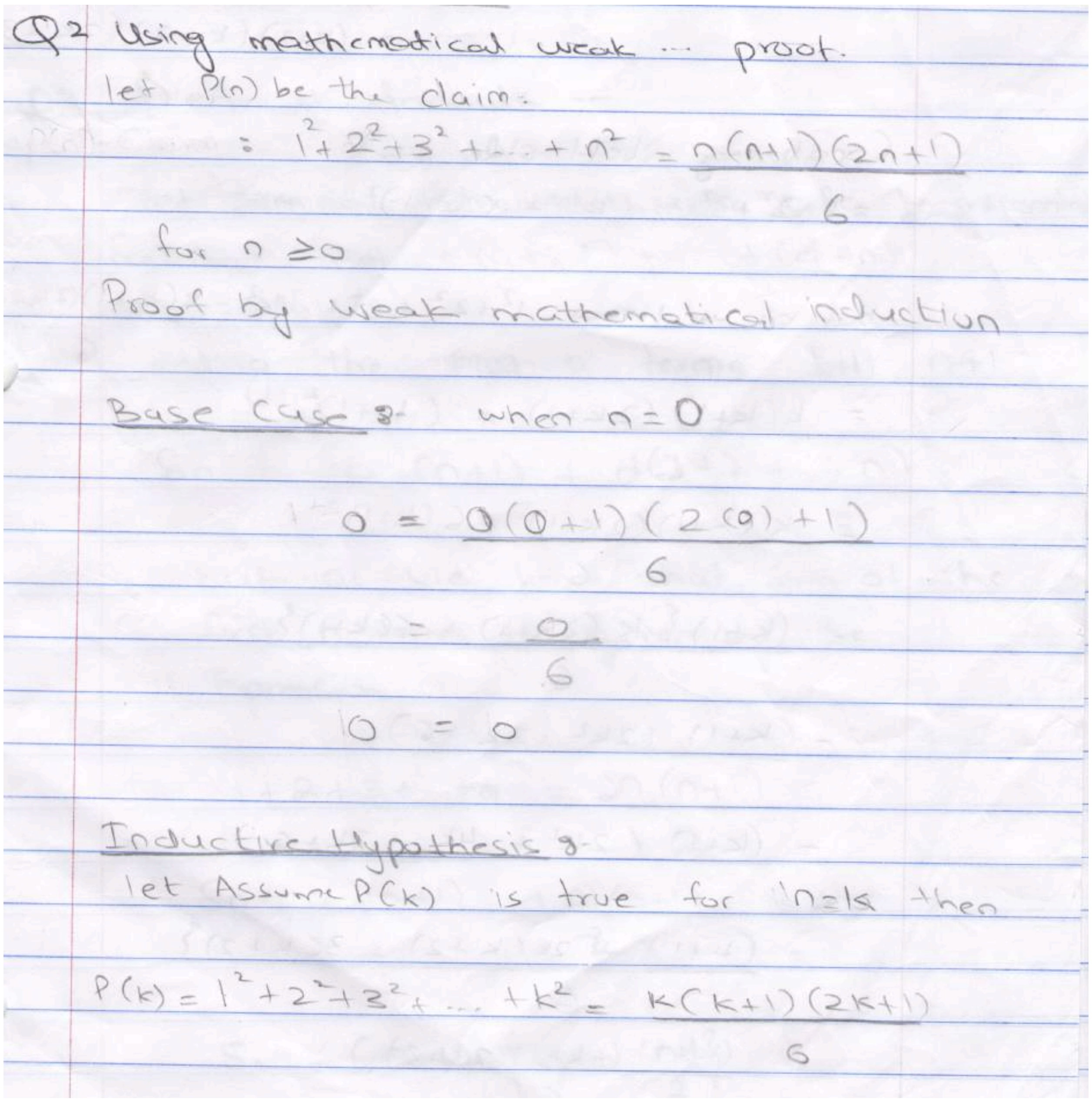
Therefore representing the sum as the following term

$$= \frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6}$$

And after simplifying in other form

$$S_n^2 = \frac{n(n+1)(2n+1)}{6}$$

2. **Using mathematical (weak) induction, prove that the formula obtained in the previous question works for all cases n ≥ 0. (If you were unable to derive the formula above, find the formula on-line so you can still do the proof.)**



Q2 Using mathematical weak ... proof.

let P(n) be the claim:

$$: 1^2+2^2+3^2+\dots+n^2 = \frac{n(n+1)(2n+1)}{6}$$

for $n \geq 0$

Proof by weak mathematical induction

Base case :- when $n=0$

$$0 = \frac{0(0+1)(2(0)+1)}{6}$$

$$= \frac{0}{6}$$

$$0 = 0$$

Inductive Hypothesis :-

let Assume P(k) is true for $n=k$ then

$$P(k) = 1^2+2^2+3^2+\dots+k^2 = \frac{k(k+1)(2k+1)}{6}$$

Inductive Step

Prove the statement holds true for $n = k+1$

$$1^2 + 2^2 + 3^2 + \dots + k^2 + (k+1)^2 = \frac{(k+1)(k+1+1)(2(k+1)+1}{6}$$

$$= \frac{(k+1)(k+2)(2k+3)}{6}$$

Considering left side i.e

$$= 1^2 + 2^2 + 3^2 + \dots k^2 + (k+1)^2$$

but as we know $1^2 + 2^2 + 3^2 + \dots k^2 = \frac{k(k+1)(2k+1)}{6}$

$$= \frac{k(k+1)(2k+1)}{6} + (k+1)^2$$

$$= \frac{k(k+1)(2k+1) + 6(k+1)^2}{6}$$

$$= \frac{(k+1)\{k(2k+1) + 6(k+1)\}}{6}$$

$$= \frac{(k+1)(2k^2 + 7k + 6)}{6}$$

$$= \frac{(k+1)(2k^2 + 4k + 3k + 6)}{6}$$

$$= \frac{(k+1)\{2k(k+2) + 3(k+2)\}}{6}$$

$$= \frac{(k+1)(k+2)(2k+3)}{6}$$

$$= \frac{(k+1)\,((k+1)+1)\,(2(k+1)+1)}{6}$$ ✓

As the proposition is true for $n=k+1$.
Therefore this states that given proposition is true as proved by mathematical Induction

Q3 Derive a formula --

let sum of the arithmetic series be represented
$S_n = a + (a+d) + (a+2d) + \ldots + (a+nd)$

adding the first a terms till $n+1$

$$S_n = a\,(n+1) + d(1+2+\ldots n)$$

but as we know that sum of the first n terms is given by the expression

$$1+2+3+\cdot\cdot+n = \frac{n(n+1)}{2}$$

$$S_n = a\,(n+1)\ \ \frac{d\,n\,(n+1)}{2}$$

$$S_n = (n+1)\left\{a + \frac{nd}{2}\right\}$$ ✓

Q4 Derive a formula for Geometric Series

let the geometric series be represented as

$$g(n) = c + cr + cr^2 + \dots cr^n \quad -①$$

Multiply $g(n)$ by r

$$r^* g(n) = cr + cr^2 + \dots cr^{n+1} \quad -②$$

Subtracting eq ① from ②

$$r^* g(n) = cr + cr^2 + \dots + cr^n + cr^{n+1}$$
$$g(n) = c + cr + cr^2 + \dots + cr^n$$
$$- \quad - \quad - \quad - \quad -$$

$$r^* g(n) - g(n) = cr^{n+1} - c$$
$$(r-1)\, g(n) = c(r^{n+1} - 1)$$
$$g(n) = \frac{c(r^{n+1} - 1)}{r-1} \quad ✓$$

Q5

let the polynomial of degree 3 be represented as

$$p(n) = an^3 + bn^2 + cn + d$$

as given values are $p(0) = 0$, $p(1) = 1$, $p(2) = 1$, $p(3) = 2$

∴ the polynomial becomes

when

$n=0, p(0):\ a(0)^3 + b(0)^2 + c(0) + d = 0 \Rightarrow d = 0$

$n=1, p(1):\ a(1)^3 + b(1)^2 + c(1) + 0 = 1$

$a+b+c = \underline{1}$ — ①

$n=2, p(2):\ a(2)^3 + b(2)^2 + c(2) + 0 = 1$

$8a + 4b + 2c = 1$ — ②

$n=3, p(3):\ a(3)^3 + b(3)^2 + c(3) + 0 = 2$

$27a + 9b + 3c = 2$ — ③

Using the wolfram tool provided

$a = \frac{1}{3}$ $b = -\frac{3}{2}$ $c = \frac{13}{6}$

∴ the given polynomial becomes

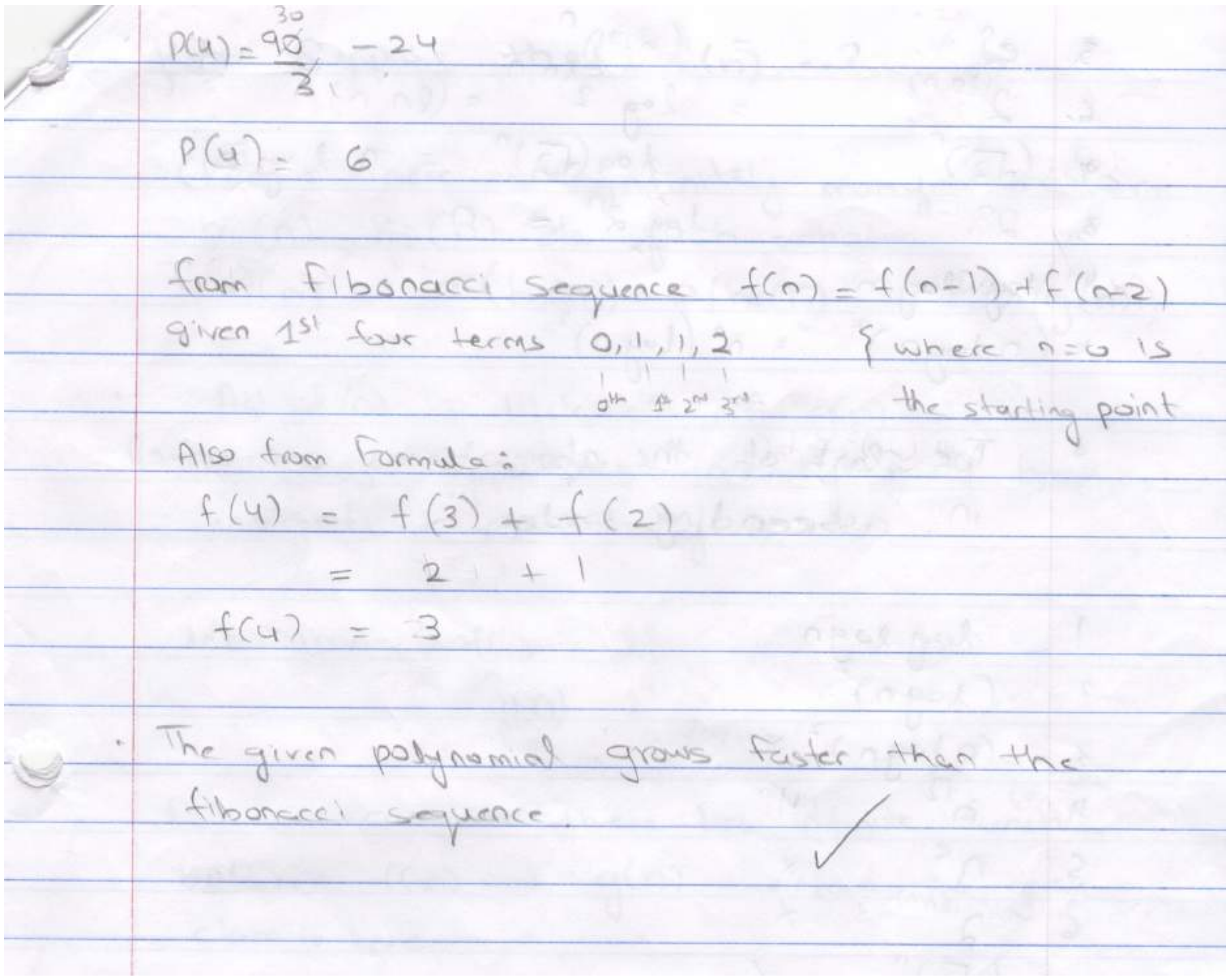
$P(n) = \frac{1}{3}n^3 - \frac{3}{2}n^2 + \frac{13}{6}n$

∴

$P(4) = \frac{(4)^3}{3} - \frac{3}{2}(4)^2 + \frac{13}{6_3}(4)^2$

$= \frac{64}{3} - \frac{3}{\cancel{2}}(4.\overset{2}{4}) + \frac{26}{3}$

$= \frac{64}{3} - 24 + \frac{26}{3}$



$P(4) = \frac{\overset{30}{\cancel{90}}}{\cancel{3}} - 24$

$P(4) = 6$

from fibonacci sequence $f(n) = f(n-1) + f(n-2)$

given 1st four terms $\underset{0^{th}}{0}, \underset{1^{st}}{1}, \underset{2^{nd}}{1}, \underset{3^{rd}}{2}$ { where $n=0$ is the starting point

Also from formula:

$f(4) = f(3) + f(2)$

$= 2 + 1$

$f(4) = 3$

- The given polynomial grows faster than the fibonacci sequence. ✓

6. **Rank these functions in increasing order of asymptotic growth: $(\log n)^2$, $n!$, $2^{(2^\wedge n)}$, n^e, e^n, $2^{(\ln n)}$, $(\surd 2)^n$, 2^n, $n^{\surd 2}$, $\log \log n$, $(n \log n)^2$**

- In order to rank the function first break them into the following groups of logarithm, polynomial, exponential, higher order terms. Then fill the function according to type. Then rank them in their respective column.

logarithm	**polynomial**	**exponential**	**Higher order**
1st log log n	3rd $2^{(\ln n)}$	7th $(\surd 2)^n$	10th n!
2nd $(\log n)^2$	4th $n^{\surd 2}$	8th 2^n	11th $2^{(2^\wedge n)}$
	5th $(n \log n)^2$	9th e^n	
	6th n^e		

7. **Suppose that f(n) = o(g(n)). Prove that there are infinitely many functions $h_1(n)$, $h_2(n)$, etc. such that f(n) = o(h_1(n)) = o(h_2(n)) = … = o(g(n)). [All the o's in this question are little-ohs.] Hint: how can you construct a new function that has an order of growth strictly between two other orders of growth?**

- Let h(n) be the function defined as
  h(n) = [g(n) / f(n)]
- When we take log of the above h(n) function we can see that it will produce yet another function that is asymptotically bigger than f(n) but smaller than g(n)
  log [g(n) / f(n)]
- as we take log again will produce asymptotically smaller function than the above function but yet still bigger than the f(n) function
  log log [g(n) / f(n)]
- Thus in this way this creates an infinite sequence of function each time that is asymptotically bigger than f(n) less than g(n).

Characteristic Equations:

let F_n to be a characteristic polynomial r^n

$$F_n = F_{n-1} + F_{n-2}$$
$$r^n = r^{n-1} + r^{n-2}$$
$$r^2 - r - 1 = 0$$
with the roots
$$r_{1,2} = \frac{1 \pm \sqrt{(-1)^2 + 4}}{2} = \frac{1 \pm \sqrt{5}}{2}$$

the general solution to equation $F_n = r^n$ will be all possible combinations of roots r_1 and r_2:

$$F_n = A\left(\frac{1+\sqrt{5}}{2}\right)^k + B\left(\frac{1-\sqrt{5}}{2}\right)^k$$

Constants A and B can be found from the boundary conditions F_0=0 and F_1=1:

$$F_0 = A + B = 0$$

$$F_1 = A\left(\frac{1+\sqrt{5}}{2}\right) + B\left(\frac{1-\sqrt{5}}{2}\right) = 1$$

$$A\left(1+\sqrt{5}\right) - A\left(1-\sqrt{5}\right) = 2$$

$$2A\sqrt{5} = 2$$

$$A = \frac{1}{\sqrt{5}}, B = -\frac{1}{\sqrt{5}}$$

The final equation is represented as:

$$F_n = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$$

Time and Space Complexity:

- The time and space complexity are defined as follow
  - Best case
    - The best case of search algorithm is when it takes the least amount of time to find the element.
  - Average case
    - average case for search algorithm is when it is neither near the lower bound of best, nor it is at the upper bound of worst, thus lying in between.
  - Worst case
    - The worst case of search algorithm is when it has to iterate through the entire data structure and find the element in the end thus taking the most amount of time.

Abstract data type and data structure:

- An abstract data type (ADT) is the realization of a data type as a software component.
- ADT does not specify how the data type is implemented
- A data structure is an implementation of abstract data type.
- Operation associated with ADT are implemented with a member function also known as method

Fundamental data structures

- **Stacks**
  - Definition: The stack is a list-like structure in which elements may be inserted or removed from only one end Queues

Stack interface (stack ADT) includes the following method

Method name	**function**
boolean isEmpty()	return true if the Stack is empty
void push(E object)	add object to the top of the Stack
E pop()	remove and return the item from the top of the Stack (error if the Stack is empty)
E peek()	return the item that is on the top of the Stack, but do not remove it (error if the Stack is empty

- **Implementation of stack**
    - The stack is implemented using the following two way
        - Array based stack implementation
        - Linked List based stack implementation

- **Queue**
    - <u>Definition:</u> A Queue is a First-In-First-Out (FIFO) abstract data type

Queue ADT includes the following method

Method Name	**Function**
public void clear()	Reinitialize the queue
public void enqueue(E it)	Place an element at the rear of the queue
public E dequeue()	`Remove and return element at the front of the queue. @return The element at the front of the queue`
public E frontValue()	return The front element.
public int length()	return The number of elements in the queue

- **Implementation of Queue:**
    - The queue can be implemented using the following way
        - Array based queue implementation
        - Circular queue
        - Linked queue

- **List**
    - <u>Definition:</u> List is a finite, ordered sequence of data items known as elements
        - Ordered mean that each element has a position
        - Each list element has a data type

    - The interface (list ADT) has the following methods

Method name	**Function**
public void clear()	Remove all contents from the list
public void insert(E item)	Insert an item at the current location
public void append(E item)	Append an element at the end of the list
public E remove()	Remove and return the current element
public void moveToStart()	Set the current position to the start of the list
public void moveToEnd()	Set the current position to the end of the list
public void prev()	Move the current position one step left
public void next()	Move the current position one step right
public int length()	return The number of elements in the list
public int currPos()	return The position of the current element
public void moveToPos(int pos)	param pos The position to make current
public E getValue()	return The current element

- **Implementation of List:**

  The list can be implemented by the following way
  - Array based implementation of list
  - Linked-list based implementation

- **Dictionary**
  - Definition: The dictionary ADT provides operations for storing records, finding records, and removing records from the collection.

the dictionary ADT includes the following method

Method Name	**Function**
public void clear()	Reinitialize dictionary
public void insert(Key k, E e)	Insert a record
public E remove(Key k)	Remove and return a record
public E removeAny() public E find(Key k)	Remove and return an arbitrary record from dictionary return A record matching "k" (null if none exists). If multiple records match, return an arbitrary one.

Implementation of dictionary:

- The dictionary implementation can be done using the following data structures
  - unsorted array-based list

- **Trees**
  - Definition: tree is a widely used abstract data type (ADT)--or data structure implementing this ADT--that simulates a hierarchical tree structure, with a root value and subtrees of children with a parent node, represented as a set of linked nodes.
  - Types: there are two main types of trees
    - Binary tree : Node having 2 children
    - N-ary tree : Node having any arbitrary no. of children
  - Application: storing data that makes it easy and searchable

  - **Tree rotation:**
    - One of the main operation of a tree is rotation also regarded as self balancing. The reason of using rotation is to not let height of the tree increases in a particular node
  - **Tree traversal**
    - refers to the process of visiting (checking and/or updating) each node in a tree data structure, exactly once. The traversal is classified in the order they visit node. Most important tree algorithms are
    - Depth first search
      - as the search tree is deepened as much as possible on each child before going to the next sibling.
      - The next table shows how the nodes are visited.
      -

Algorithm technique	**Node Visited as**
Pre-order	Root, preorder (left), preorder(right)
Post-order	Postorder (left), postorder(right), root
Inorder	Inorder(left), root, inorder(right)
Reverse order	Reverorder(right), root , reverorder(left)

    - Breadth first search
      - as the search tree is broadened as much as possible on each depth before going to the next depth.

  - **Implementation of Tree**
    - There are implementation of tree
      - The array based implementation
      - As can be seen e.g. of storing binary tree on array based implementation first root take place (pos: 1) then its two children (pos: 2,3) and then its children.

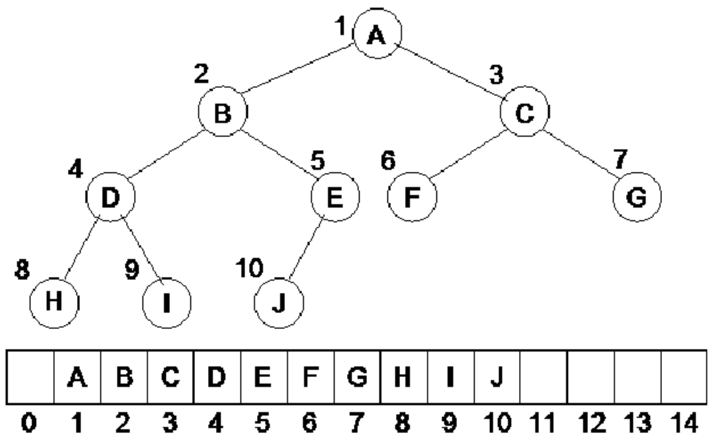


Figure 29.1: Complete Binary Tree

▪

- The linked list based implementation

- **Heap**
  - Heap is a data structure that satisfy the heap property. The heap property states that the lowest element is stored in top of the heap. Heap is partially sorted.
  - Method present in the heap ADT is shown below

Method name	**function**
Find-max	Find the maximum item of the heap
Insert	Add a new element to the heap
Extract-min	Find the minimum element of the heap which is the root element.
Delete-max or Delete-min	Delete the maximum or minimum element of the heap

  - Heap is also called priority queue because the priority element is always at the top of the data structure

Next Class: Sorting Algorithms

References:

1. http://people.cs.vt.edu/shaffer/Book/JAVA3elatest.pdf
2. https://en.wikipedia.org/wiki/Tree_%28data_structure%29
3. https://en.wikipedia.org/wiki/Tree_traversal
4. https://en.wikipedia.org/wiki/Heap_%28data_structure%29
5. http://ulcar.uml.edu/~iag/CS/Fibonacci.html