DATE: __/__/____
PAGE: _____

# lecture 3

## Stack

- Last In, First Out (LIFO)

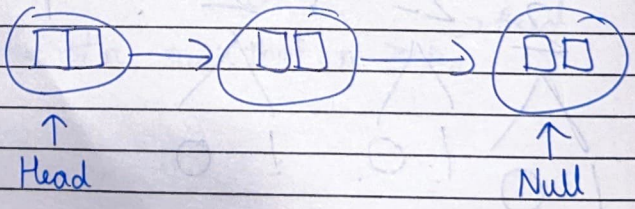
## Queue

- First In, First Out (FIFO)

## Priority Queue / Heap

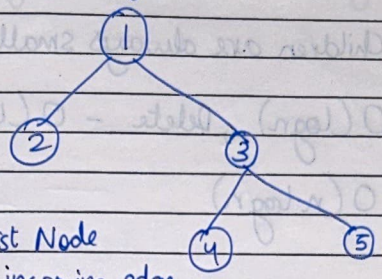
- Normal Queue, Max gets removed

## Linked List



DATE: __/__/____
PAGE: _____

# Binary Tree




Root - first Node
Child - An incoming edge
Sibling - Same parent

Degree - No. of children of a particular parent
Level - Distance from Root
Height -

## Complete Binary Tree

All the levels of the tree are filled completely except the lowest level nodes, which are filled from as left as possible.

- Very efficient while implementing an array

Parent : $i$ → left child : $2i+1$
Right child : $2i+2$

Child : $j$ → Parent : $\left\lfloor \frac{j-1}{2} \right\rfloor$

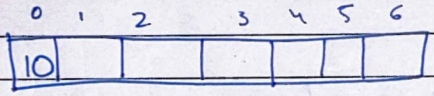
DATE: __/__/____
PAGE: _____

# MaxHeap (Height = log n)

- Constraint - Children are always smaller
- Insert - O(logn), Delete - O(logn)
- Sorting - O(n logn)

Eg.

10 12 2 13 17 1 19



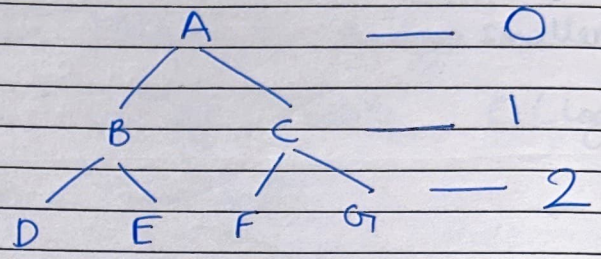

(10)

Most No. of Swaps = log (n)

Creating a Heap = [illegible]

☆ Explain Heapify Algorithm being O(n)
☆ Explain Heap Sort is O(n logn)

# Heaps




In an Array : $\underset{0}{[A}, \underset{1}{B}, \underset{2}{C}, \underset{3}{D}, \underset{4}{E}, \underset{5}{F}, \underset{6}{G}]$

If parent = i, children = $2i+1$, $2i+2$
If child = i, parent = $\left\lfloor \frac{i}{2} \right\rfloor$

Eg. Index of A = 0
Index of Children = $2(0)+2$, $2(0)+1$
$= 1, 2 = B, C$

Index of F = 5
Index of Parent = $\left\lfloor \frac{5}{2} \right\rfloor = 2 = C$

Height of tree = h, Nodes = $2^{h+1}$ (Maximum)

Good Write

# Pseudocode for Building a Heap

```
heapInsert(arr, n, element):

    i = n
arr[n] = element
while (i ≠ 0):
    ~~if (arr[i] > arr[i/2]~~
    parent = (i-1)/2
    if (arr(parent) < arr(i)):
        swap(&parent, &i) → Give memory location
    else:
        break
    i = parent


heapify(arr, n, i):

while true:
    left = 2i+1  → Child
    right = 2i+2 → Child
    largest = i  → Assuming our node is Max

if (left < n AND arr[left] > arr[largest])
    → largest = left

if (right < n AND arr(right) > arr(largest)
    → largest = right
```