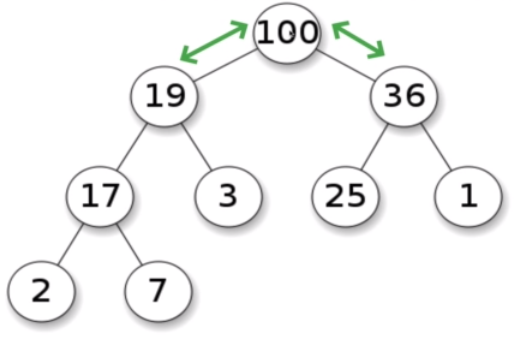
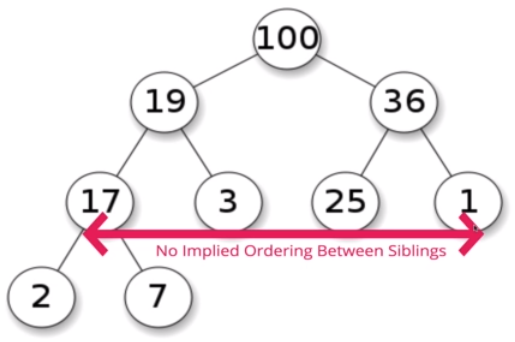
JavaScript Algorithms and Data Structures Masterclass

# Section 24: Binary Heaps

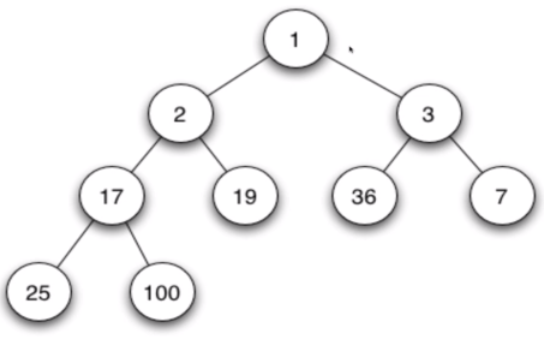
## Intro to Binary Heaps

* Similar to Trees, but not ‘ordered’.
  + Only that parents and nodes relationship by size
  + Binary Heaps are used to implement **Priority Queues** and **Graph Traversal Algorithm**

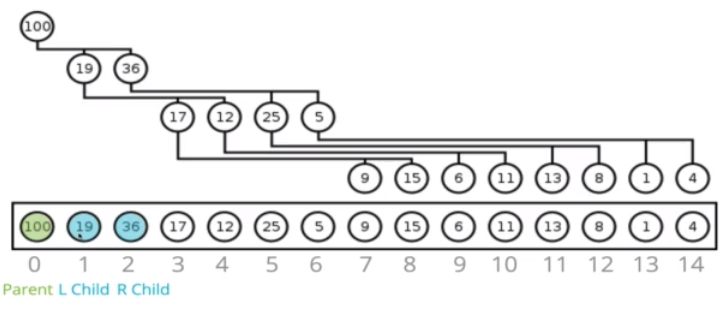
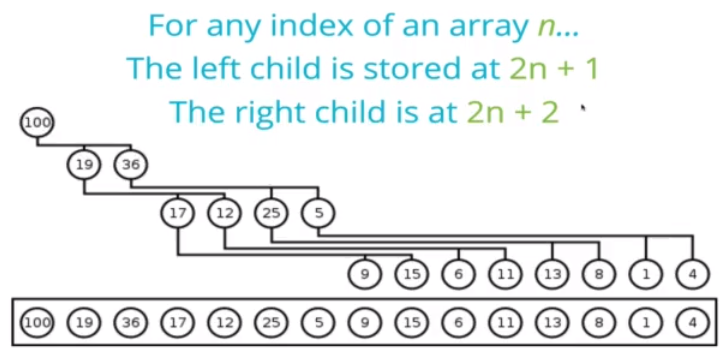
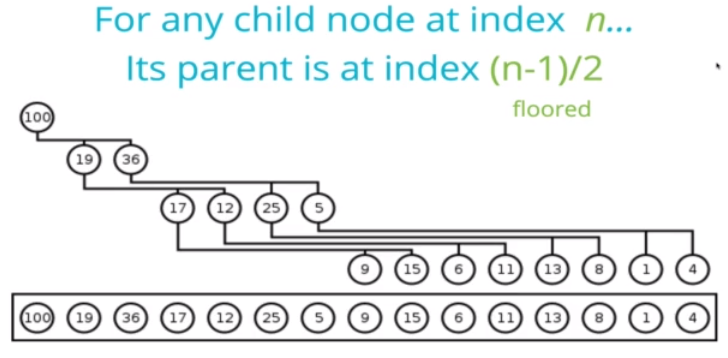
### Max Binary Heap

* Parent Nodes > its Child Nodes (max 2) \*\*\***Root is the biggest number!**
* **Rules**:
  + Each Parent node has max 2 Children nodes
  + Value of the Parent is **ALWAYS** > Child node
  + Sibling nodes (same level) aren’t guaranteed to be greater.. ONLY Parent is bigger!
  + Binary Heaps are as compact as possible (takes least amount of space as possible)
    - Children of each node are as full as they can, so you must fill in the children spaces before you can move down level
    - Left Children are filled in first!
* *Relationships in Max Binary Heap*
  +  

### Min Binary Heap

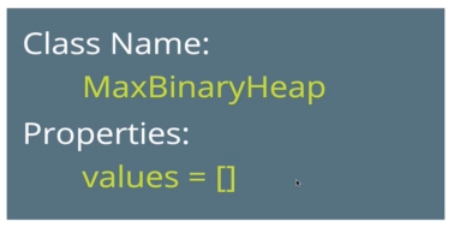
* Parent Nodes < its Child Nodes
  + Example of a Min Binary Heap
    - 

## Storing a Binary Heap

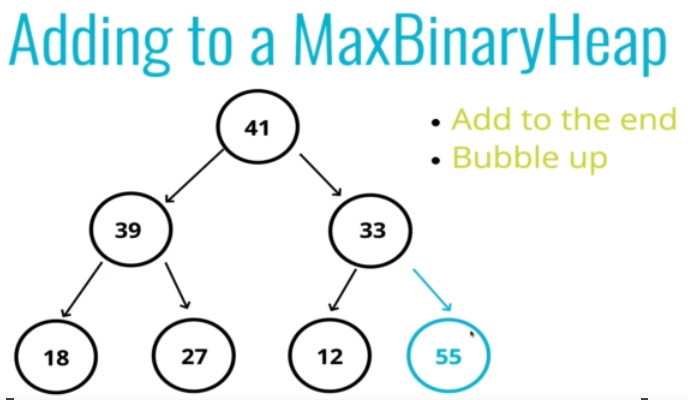
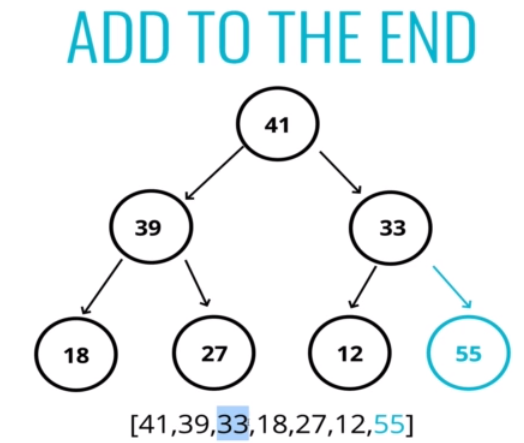
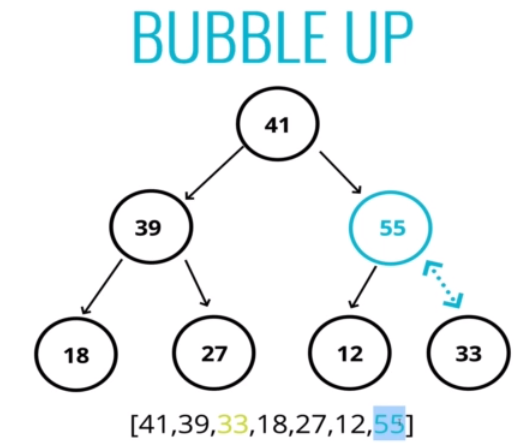
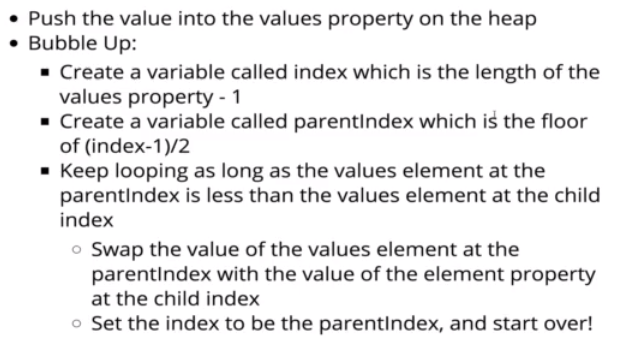
* You can use an Array or List
  + **Ex**. 
* **Rule**:
  + Finding Parent’s Children Nodes in the array
    - 
  + Finding Child Node’s Parent in an array
    - 

## Building a Binary Heap

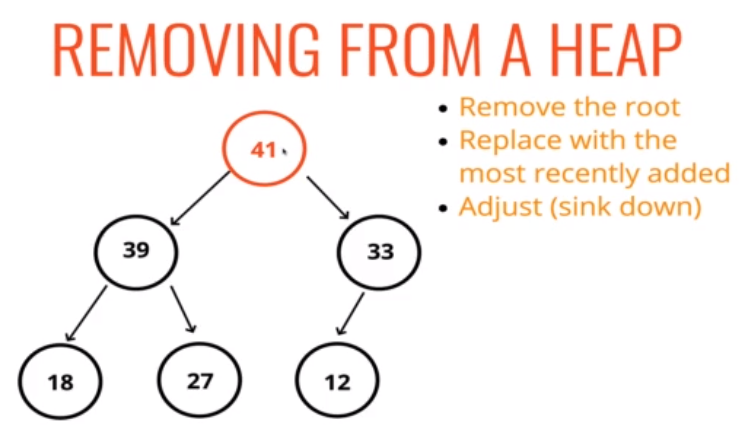
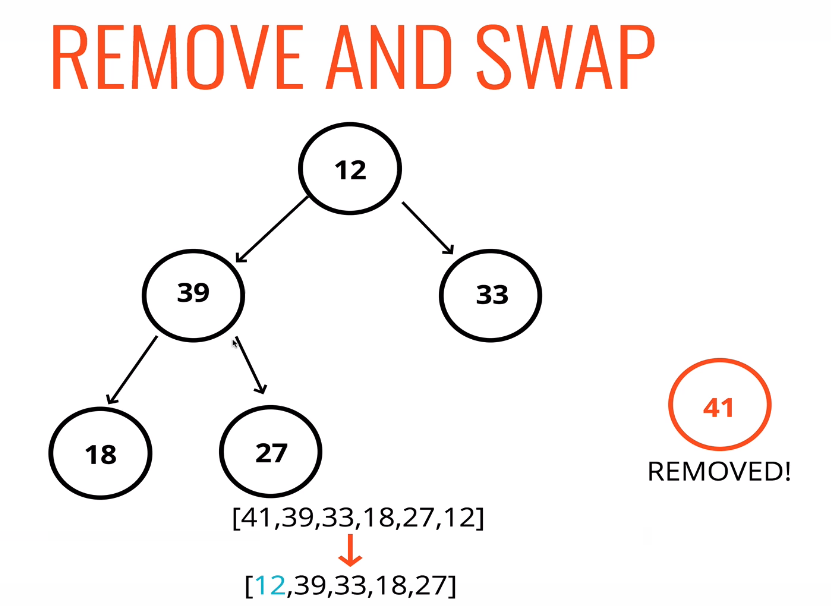
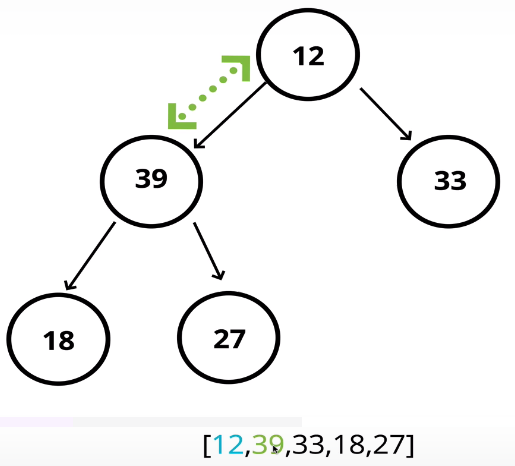
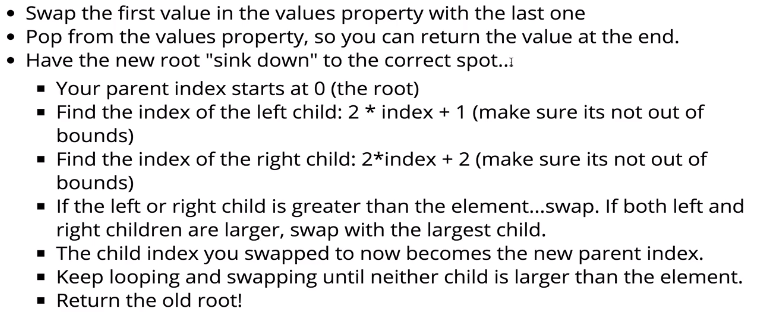
### Defining the Class

* 

### Insert()

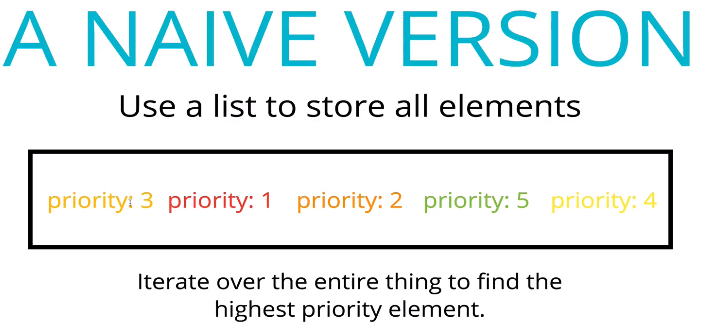
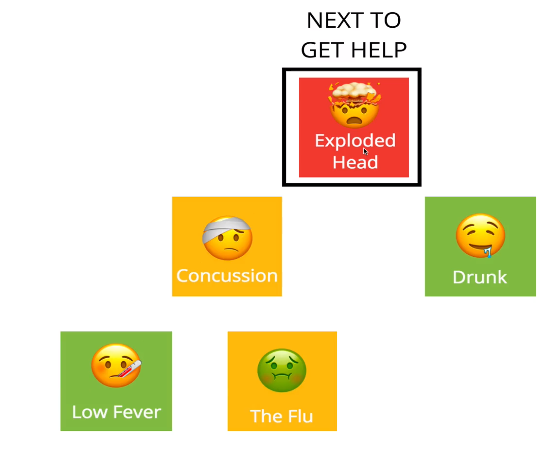
* 
*  
* **Psuedo-code**:
  + 
    - Push first to the end of array, then bubble up

### ExtractMax()

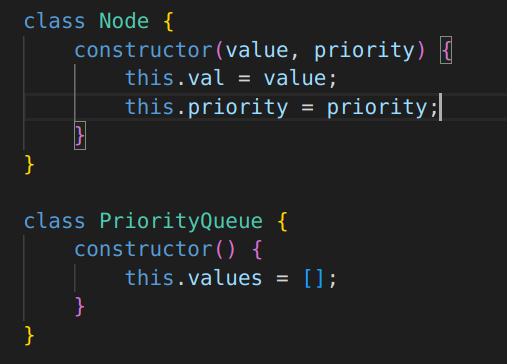
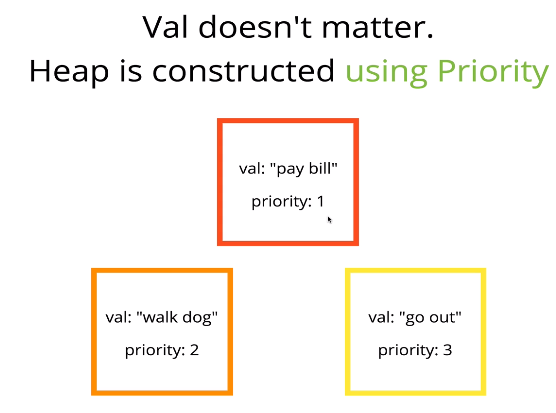
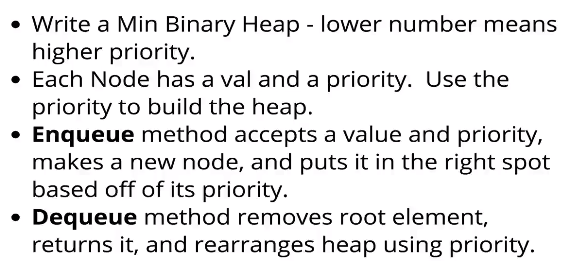
* AKA Removing Max Value (root) from a Binary Heap
  + 
  + **2 Part Operation**:
    - Remove the Root & Replace root with newest element
      * 
    - **Sink-down**
      * This operation for deleting the root from Binary Heap (extracting either MIN or MAX element) and restoring the properties is called *down-heap* 
        + **aka**: bubble-down, percolate-down, sift-down, trickle-down, heapify-down, cascade-down, extract min/max
        + 
        + Compare the newly added element with its children (swap with the largest child) and then move to next level and repeat
* **Psuedo-code**:
  + 

## Priority Queues

### Intro to Priority Queues (PQs)

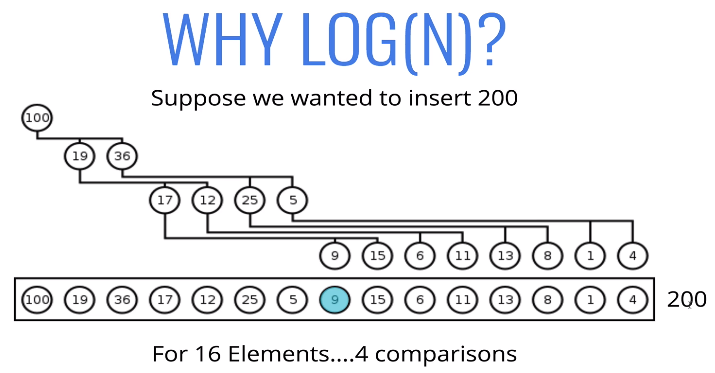
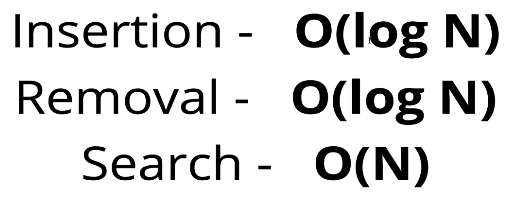
* A DS where each element has a priority
  + Higher priority elements are served before Lower priority elements
* **Naive Approach – List**
  + ****
* **Efficient Approach – Heap**
  + ****
    - **Inserting**:
      * Add from the bottom of the heap, then bubble up the higher priority elements
    - **Remove**:
      * Pop off the top, then bubble up the next highest priority element to be the new root

### Implementing a Priority Queue (Min Binary Heap – lower priority served first)

* **Classes**:
  + 
* **Node**:
  + 
* **Psuedo-Code**
  + 
    - Enqueue = Insert
    - Dequeue = Remove

## Big O of Binary Heaps

### Time Complexity

* **How many comparisons?**
  + 
    - Log base 2
* 
* **Recap**:
  + 