Data Structures

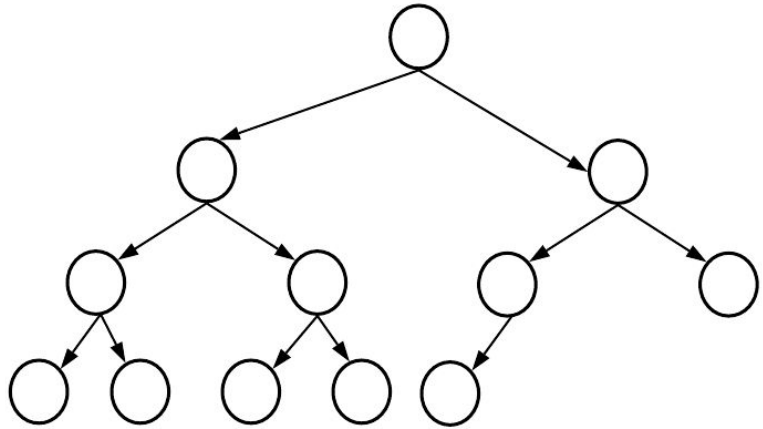
Binary Heap

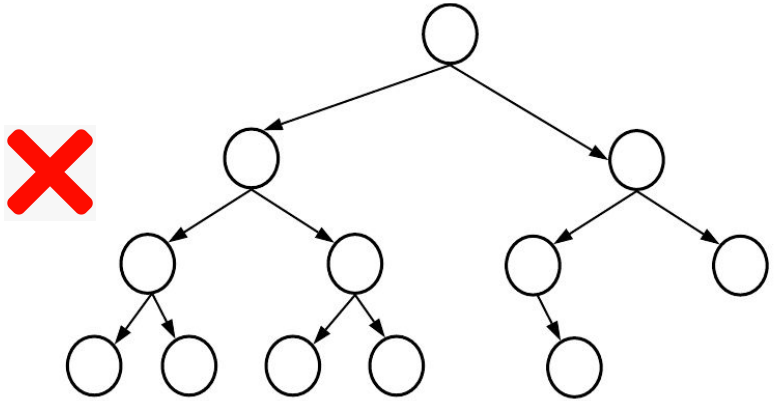
Mostafa S. Ibrahim
Teaching, Training and Coaching for more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

Recall: Complete Binary Tree

- All levels are **complete**
  - With the **possible** exception of the last level, which must be filled from the **left**!
- Top tree
  - 4 levels
  - The first 3 are complete
  - The last level is filled from the left
- Bottom tree: NOT complete
  - A right node has been filled before a left one
- Height = ceil($\log_2(n+1)$) - 1
  - Each complete level is 2^i nodes



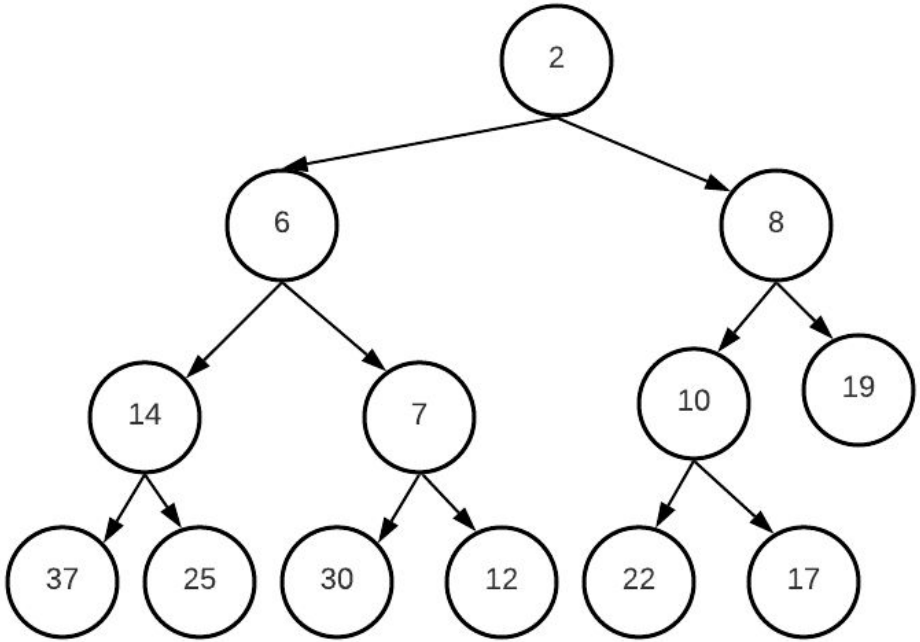


Binary Tree, Binary Search Tree and Binary Heap

- Binary Tree: max 2 children per node - simple structure
  - But search is O(N)
- We need to find a faster search structure!
- BST root: left < root < right
  - We can search in O(h), which is great IFF the tree is **balanced**
- In many cases we need to find the **min**(or max) node and **delete** it fast!
  - This is where the Binary Heap proves to be very useful!

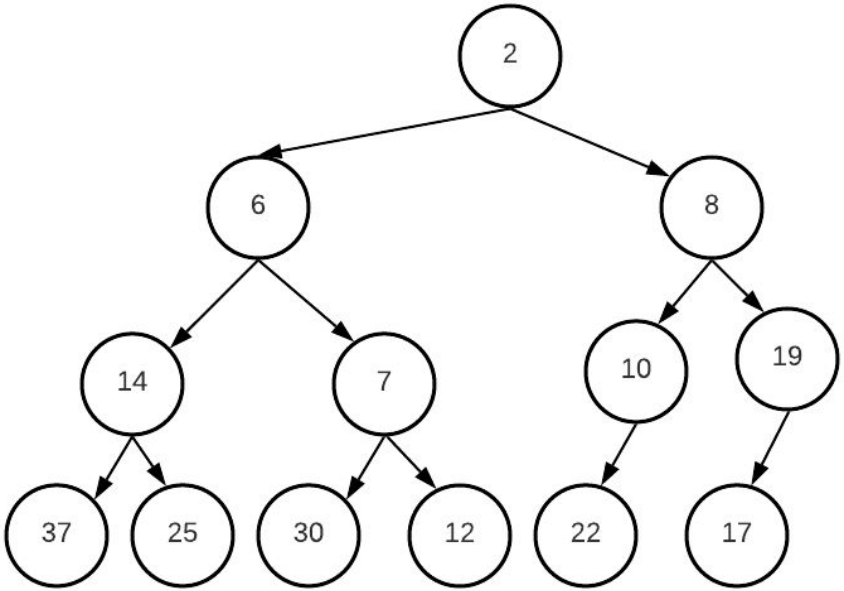
What is (min) (binary) heap?

- A **complete** binary tree where any node <= **ALL** its children.
- Hence: Root has the minimum value in the tree!
- A max heap has exactly the opposite definition
  - >= all its children
- Note: whenever I use the word 'heap' in these lectures, I'm talking about a MIN BINARY HEAP



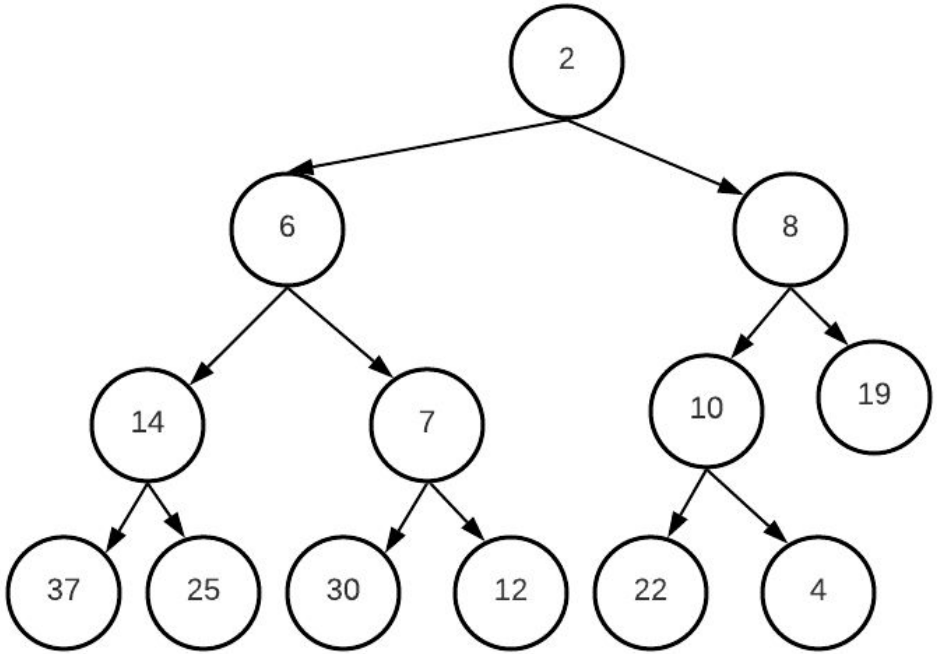
Not a min heap tree

- Not a complete tree
  - Node 10 doesn't have a right child
  - To qualify as a min heap tree, given the lack of a right child on node 10, there must be NO further nodes on the 'child' level from this point



Not a min heap tree

- A complete tree, but node(10) is not smaller than its 2 children!
  - 10 > 4



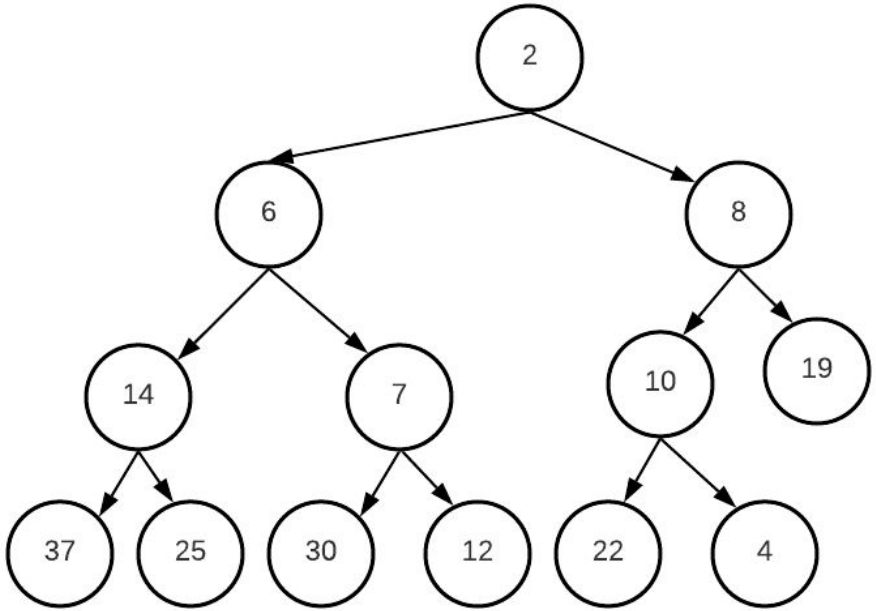
Heap ADT

- top() refers to the **min value** in the heap
- pop() will **remove the smallest** value from the heap
- Otherwise, the push() and pop() functions work just like they do in a queue
- We can print the content using isempty(), top(), and pop()

```
class MinHeap:
    def __init__(self):...

    def pop(self):...

    def push(self, key):...

    def top(self):...

    def empty(self):...


if __name__ == "__main__":
    minHeap = MinHeap()
    for val in [2, 17, 22, 10, 8, 37, 14,
            19, 7, 6, 5, 12, 25, 30]:
        minHeap.push(val)

    while not minHeap.empty():
        print(minHeap.pop(), end=', ')
    # 2, 5, 6, 7, 8, 10, 12, 14, 17,
    # 19, 22, 25, 30, 37
```

Your turn: Think for 10 min (for each)

- We learned to code a binary tree based on pointers
- Complete binary trees can be represented using arrays in an interesting way. How?
  - Recall the number of nodes per level
- Assume we have this min heap: how to insert value (5)?
  - Tip: Add it to the next available node (left of 19), then **fix** the tree branch!



"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."