BCSE202L Data Structures and Algorithms

Heaps

Overview

Binary Trees Revisited

Heaps

Maintaining Heap Property

Building a Max Heap - Top Down Approach

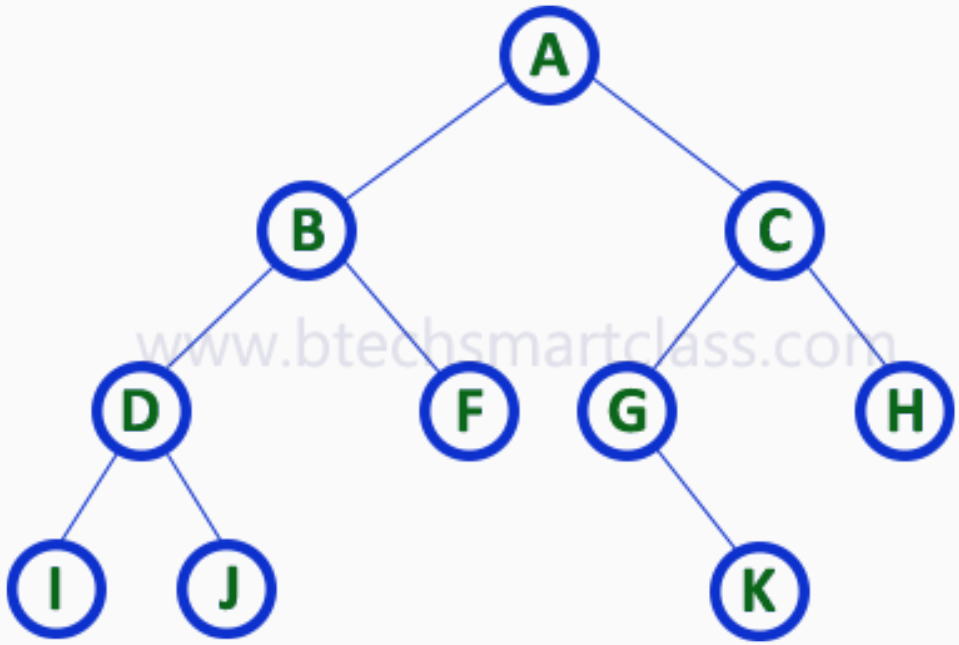
Building a Max Heap - Bottom Up Approach

Operations on Heap

Heap Sort

Binary Trees Revisited

Binary Trees: Array Representation



A	B	C	D	E	G	H	I	J				K
0	1	2	3	4	5	6	7	8	9	10	11	12

Deduction of Parent-Child Relationship

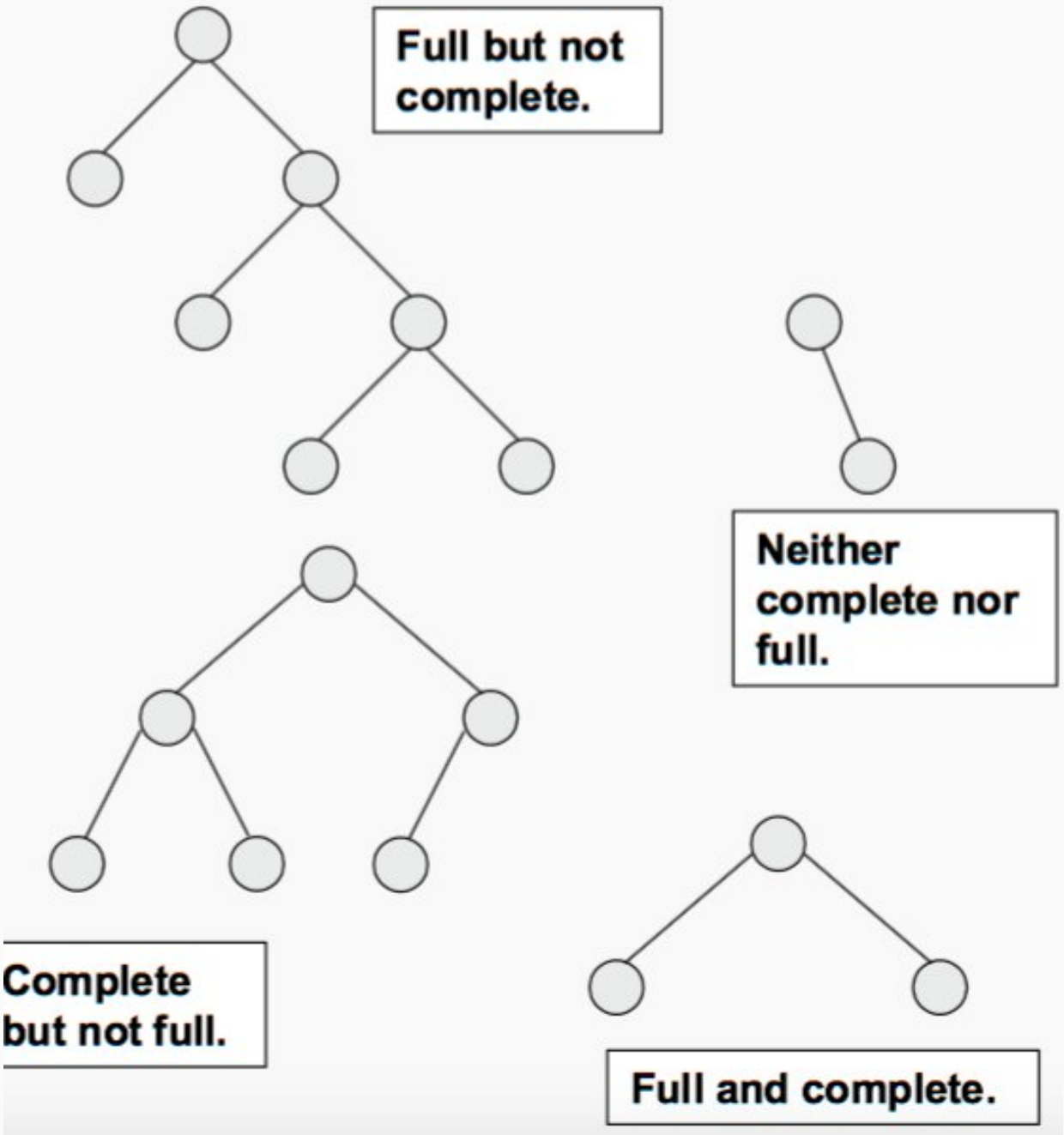
- It is easy to deduce from a plotted tree
- How to deduce from actual implementation (array representation)?
- For any node at index i (Assuming indices start with 0)
  - Left child would be at $2i + 1$
  - Right child would be at $2i + 2$
  - Parent would be at $\lfloor \frac{i-1}{2} \rfloor$

Full/Proper/Strict Binary Tree

- Every node has either 0 or 2 children.
- In other words, all nodes except leaf nodes have exactly 2 children
- There won't be gaps in array representation

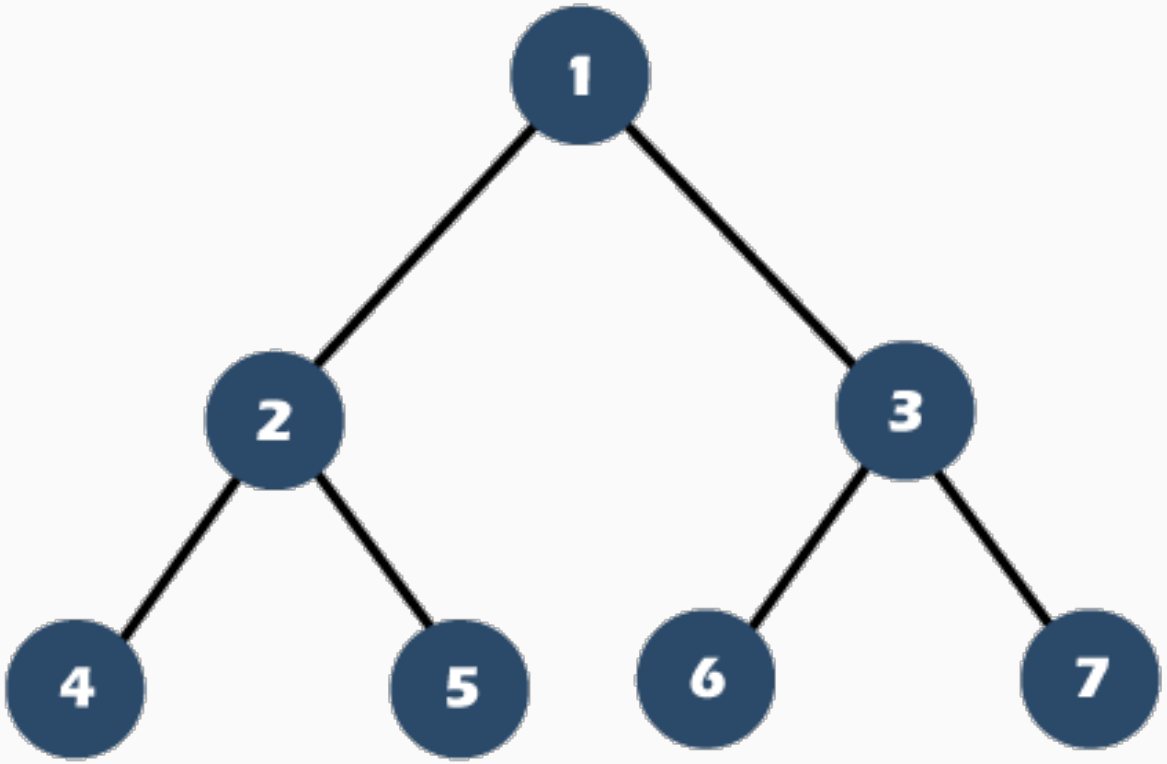
Complete Binary Tree

- All levels are completely filled (except possibly the last level). In the last level, nodes must be aligned as left as possible
- There won't be gaps in array representation
- Height of a complete binary tree of n nodes is $\log n$


Full but not complete.
Neither complete nor full.
Complete but not full.
Full and complete.

Perfect Binary Tree

- A binary tree in which all interior nodes have two children and all leaves have the same depth or same level



Heaps

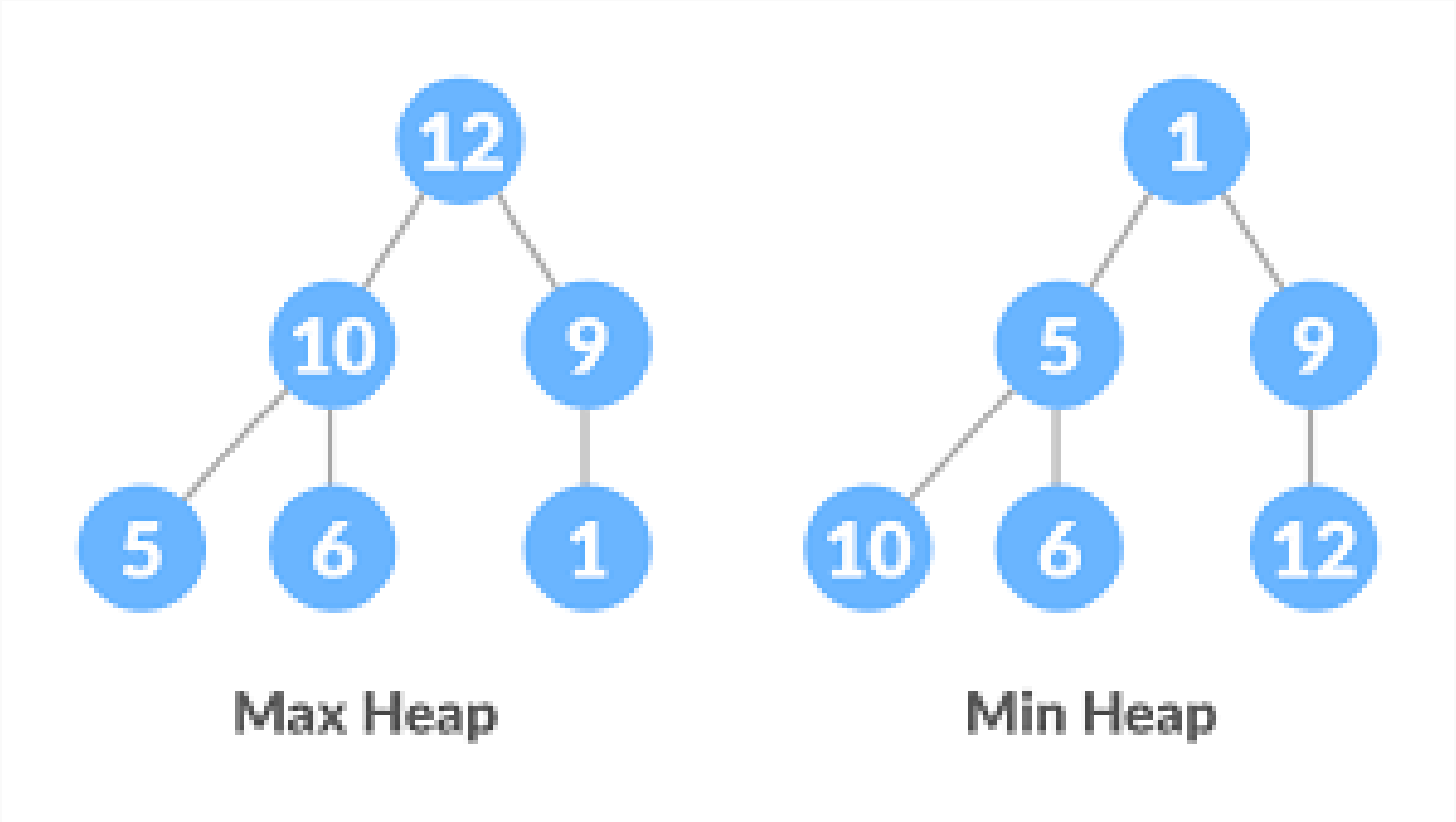
Heaps

- Heap is a tree-like data structure
- Heap is a complete binary tree
- Heap can be of two types

Max Heap and Min Heap

1. Max-Heap: In a Max-Heap the key present at the root node must be greatest among the keys present at all of it's children. The same property must be recursively true for all sub-trees in that binary tree
2. Min-Heap: In a Min-Heap the key present at the root node must be minimum among the keys present at all of it's children. The same property must be recursively true for all sub-trees in that binary tree.

Max Heap and Min Heap Example



12	10	9	5	6	1
0	1	2	3	4	5

Max Heap Array Rep.

1	5	9	10	6	9
0	1	2	3	4	5

Min Heap Array Rep.

Heap Properties in Array Representation

- for every node i
- Max Heap $\longrightarrow A[i] \geq A[\text{Child}(i)]$
- Min Heap $\longrightarrow A[i] \leq A[\text{Child}(i)]$
- *except for leaves, which have no children*
- Alternatively,
- Max Heap $\longrightarrow$ Every node value is $\geq$ its descendent value
- Min Heap $\longrightarrow$ Every node value is $\leq$ its descendent value

Maintaining Heap Property

The Heapify Method

- We will learn for max heap, but can be applied for min hap also with small changes in the symmetry
- Therefore, the function used is MAX-HEAPIFY
- We will use an array A as input

MAX-HEAPIFY(A, i)

1: $l = \text{Left}(i)$
2: $r = \text{Right}(i)$
3: **if** $l < A.heap\_size$ and $A[l] > A[i]$ **then**
4: $\quad largest = l$
5: **else**
6: $\quad largest = i$
7: **end if**
8: **if** $r < A.heap\_size$ and $A[r] > A[largest]$ **then**
9: $\quad largest = r$
10: **end if**
11: **if** $largest \neq i$ **then**
12: $\quad$ exchange $A[i]$ with $A[largest]$
13: $\quad$ MAX-HEAPIFY(A,$largest$)
14: **end if**

Building a Max Heap - Top Down Approach

Build Max Heap Top Down

- No pseudocode available in many of the standard textbooks and references
- But you know the process!!
- Write a pseudocode for this
- Will be included in DA
- Next seminar topic for volunteers

Build Max Heap: Top Down - Example

4	1	3	2	16	9	10	14	8	7
0	1	2	3	4	5	6	7	8	9

- Exercise: Illustrate the operation of BUILD-MAX-HEAP on the array $A = \langle 5, 3, 17, 10, 84, 19, 6, 22, 9 \rangle$

Building a Max Heap - Bottom Up Approach

BUILD-MAX-HEAP(A)- Bottom Up

1: $A.heap\_size = A.length$
2: **for** i $= A.length/2 - 1$ `downto 0` **do**
3: $\quad MAX - HEPIFY(A, i)$
4: **end for**

Build Max Heap: Top Down - Example

4	1	3	2	16	9	10	14	8	7
0	1	2	3	4	5	6	7	8	9

- Exercise: Illustrate the operation of BUILD-MAX-HEAP on the array $A = \langle 5, 3, 17, 10, 84, 19, 6, 22, 9 \rangle$

Analysis: Top Down - $O(n \log n)$

- A complete binary tree with n nodes will have height $\log n$
- $\therefore$ MAX-HEPIFY can take $O(\log n)$ time
- BUILD-MAX-HEAP calls MAX-HEPIFY for inserting n elements
- $\therefore$ Complexity of BUILD-MAX-HEAP is $O(n \log n)$ in the worst case if top-down approach is used

Analysis: Bottom Up - $O(n)$

- The complexity of BUILD-MAX-HEAP is $O(n)$ if bottom-up approach is used
- How? Find answer here: `https://www.youtube.com/watch?v=8noP3YjjJCM`
- **Takeaway Point: Bottom-up is better than top-down in terms of computational complexity**

Operations on Heap

Operations on Heap

1. Heapify $\longrightarrow$ Process to rearrange the heap in order to maintain heap-property
2. Find-max (or Find-min) $\longrightarrow$ find a maximum item of a max-heap, or a minimum item of a min-heap, respectively $\longrightarrow O(1)$ time
3. Insertion $\longrightarrow$ Add a new item in the heap
4. Delete - Extract Max (Extract-Min)$\longrightarrow$ Returning and deleting the maximum or minimum element in max-heap and min-heap respectively $\longrightarrow O(\log n)$ time

Extract Max or Delete from Heap

- Only the root can be deleted
- Equivalently, the element at index-0 can only be deleted in array representation
- Replace the root with the last node
- Heapify top down

$Extract - Max(A)$ Pesudocode

1: **if** $A.length < 1$ **then**
2: Error- Heap Underflow
3: **end if**
4: $max = A[0]$
5: $A[0] = A[A.length - 1]$
6: $A.length = A.length - 1$
7: $MAX - HEAPIFY(A, 0)$
8: **return** max

Deletion Examples

1. Build a max heap with the elements 4, 1, 3, 2, 16 , 9 , 10 , 14 , 8 , 7 and delete 3 times
2. Build a max heap with the elements 5, 3, 17, 10, 84, 19, 6, 22, 9 and delete 3 times

Heap Sort

Heap Sort (Ascending Order) Outline: Steps

1. Build a max heap from the given elements
2. Perform deletion one-by-one

- Demo: Heap sort 15, 20, 7, 9, 30

Heap sort Pseudocode: $HEAPSORT(A)$

1: BUILD-MAX-HEAP(A)
2: **for** i = $A.length - 1$ `downto` 1 **do**
3: exchange $A[0] \leftrightarrow A[i]$
4: $heap - size[A] = heap - size[A] - 1$
5: $MAX - HEPIFY(A, 0)$
6: **end for**

Analysis

- $O(n)$ for BUILD-MAX-HEAP(Line 1), if bottom-up approach is used
- The for loop will run for n times
- The MAX-HEAPIFY inside the loop will take $O(\log n)$ time in the worst case
- The overall complexity is $O(n) + O(n \log n) = O(n \log n)$

Summary

Binary Trees Revisited

Heaps

Maintaining Heap Property

Building a Max Heap - Top Down Approach

Building a Max Heap - Bottom Up Approach

Operations on Heap

Heap Sort

Thank you..!!