Design and Analysis of Algorithms

Lecture - 14

Success is always inevitable with Hard Work and Perseverance

N. Ravitha Rajalakshmi

Learning Objective

- Discuss Heap Sort
    - Heap , Property and its Construction

Sorting

Input: An array A with n elements.

Output: Permutation of Array A where elements are arranged in non-decreasing order.

Initial Array	130	10	40	8	20	200
Sorted Array	8	10	20	40	130	200

Heap Sort

Array Elements

130	10	40	8	20	200

- Transform and Conquer

  Change the representation of input data

200	20	130	8	10	40

Heap

- Identify and remove Maximum element recursively to find the sorted order

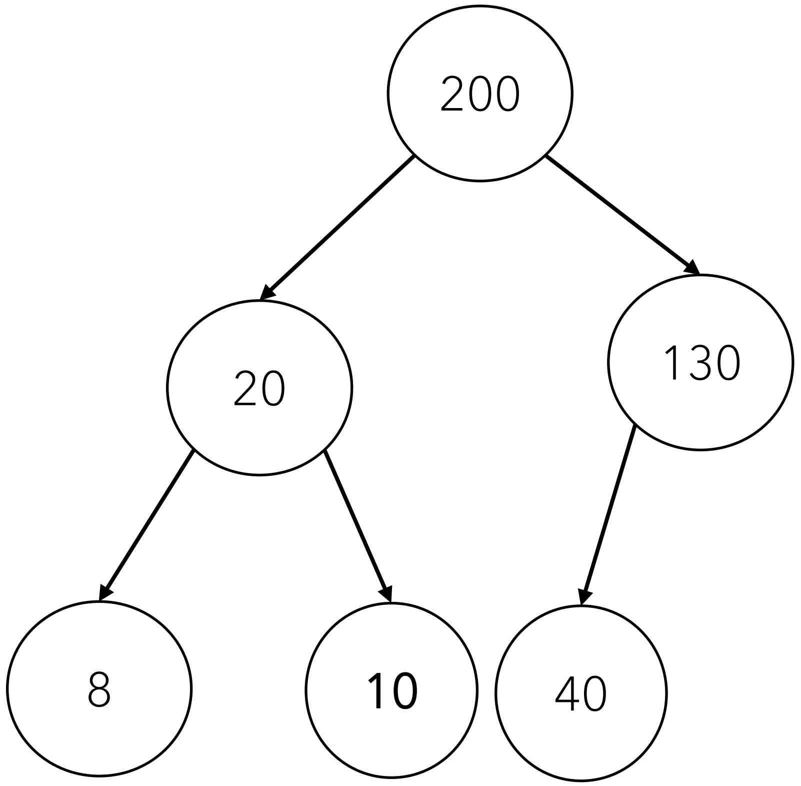
8	10	20	40	130	200

Heap

- Array object visualized as a binary tree
- Nearly complete binary tree (all levels are filled except at the last level)
- Two import attributes of heap
  - Length - Length of the entire array
  - Heap size – Number of elements in the heap

  Heap size ≤ length

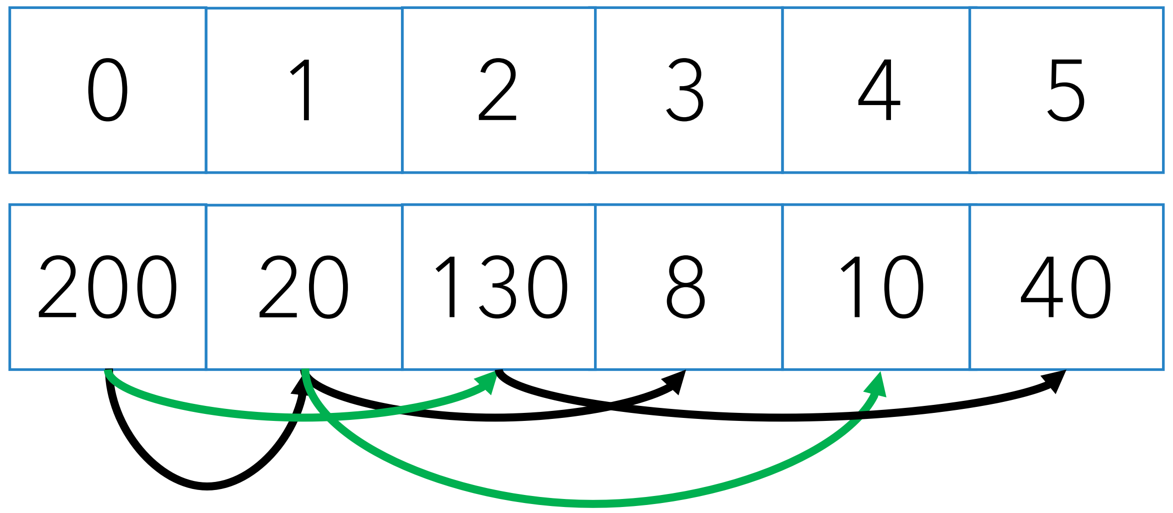
200	20	130	8	10	40



Pause & Think

- Element at 0 will be the root node in the tree
- How the elements are related?

  Given an element at i^{th} position, what will be position of parent , left child and right child



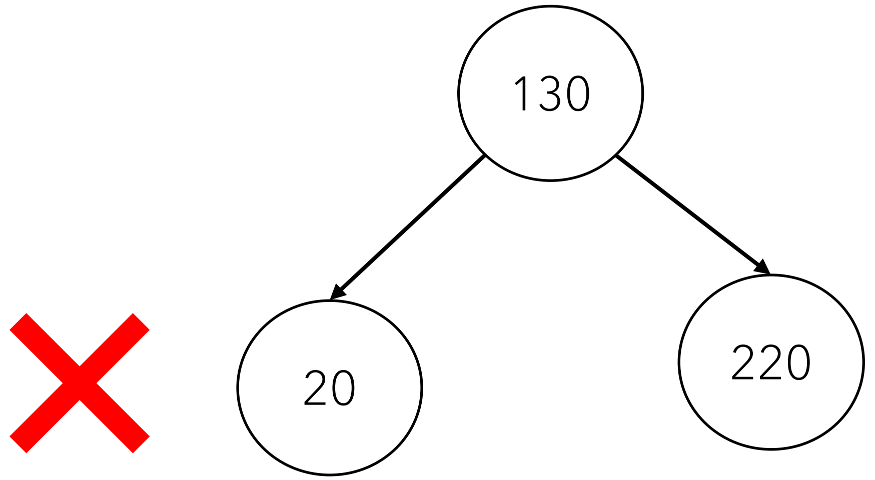
0	1	2	3	4	5
200	20	130	8	10	40

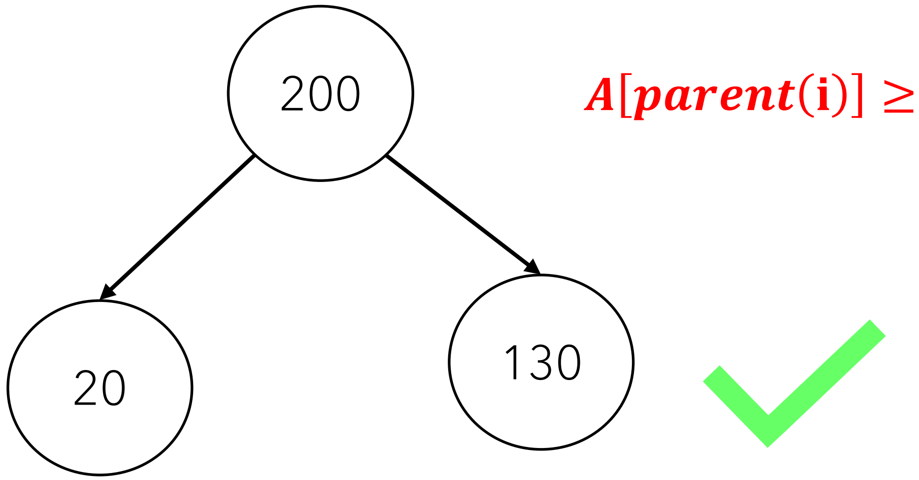
Pause & Think

Function Parent(i)	Function left(i)	Function right(i)
return (i-1)/2	*return (i*2)+1*	*return (i*2)+2*

Heap property

- Values in the nodes should satisfy heap property
- Two kinds of heaps : min heap and max heap
- Max heap – Parent should hold larger value compared to its children





$\boldsymbol{A[parent(\mathrm{i})] \geq A[\mathrm{i}]}$

Heap property

- Min heap – Parent should hold smaller value compared to its children

$$\boldsymbol{A}[\boldsymbol{parent}(\mathbf{i})] \le \boldsymbol{A}[\mathbf{i}]$$

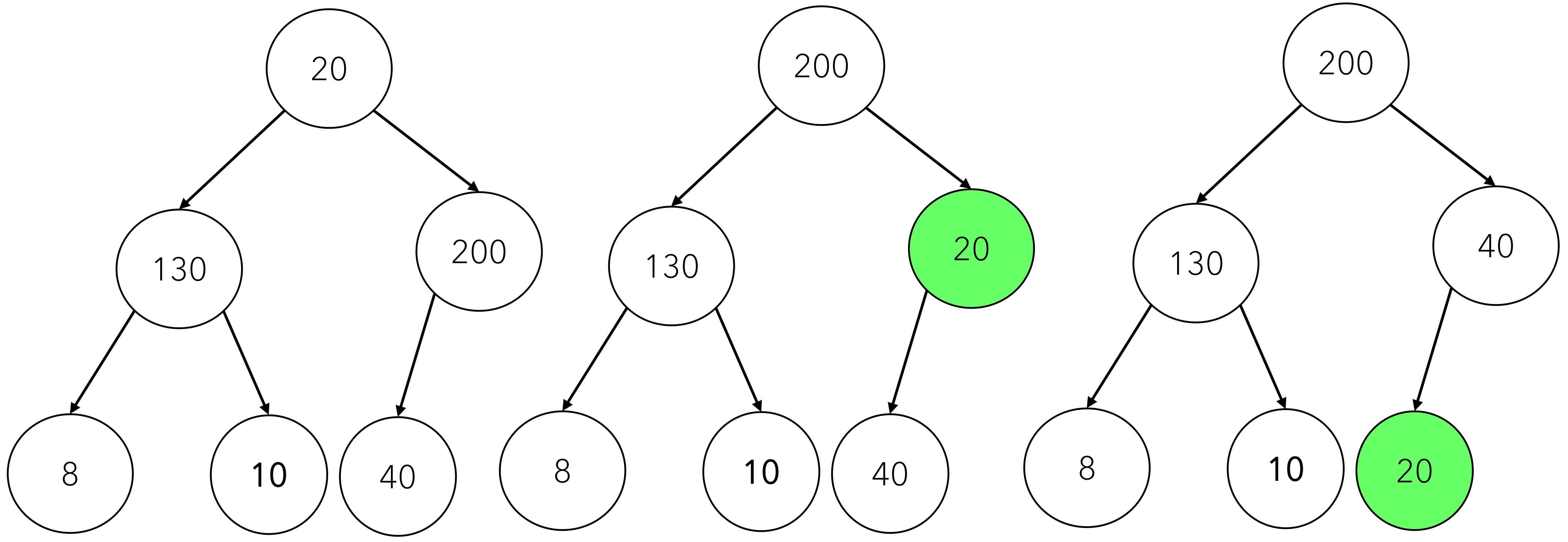
- If heap property is violated at any node in the tree, heapify procedure is used.

Max-Heapify

- When a node i violates the heap property then heapify is applied

$$\boldsymbol{A[parent(\mathrm{i})] \geq A[\mathrm{i}]}$$

- Either the right child (or) left child contains a larger value
- Swap the value of parent and child (with larger value)
- Is that sufficient??


20
130
200
8
10
40
200
130
20
8
10
40
200
130
40
8
10
20

Function Max-Heapify(i)

```
l = Left(i)
r = right(i)
gt = i  # index of largest element
if(A[gt]<A[l] && l<heapsize)
        gt = l
if(A[gt]<A[r] && r<heapsize)
        gt = r
if(gt!=i){
        swap (A[i], A[gt])
        Max-Heapify(gt)
}
```

Heap Sort

Array Elements

130	10	40	8	20	200

- Transform and Conquer

  Change the representation of input data

200	20	130	8	10	40

Heap

- Identify and remove Maximum element recursively to find the sorted order

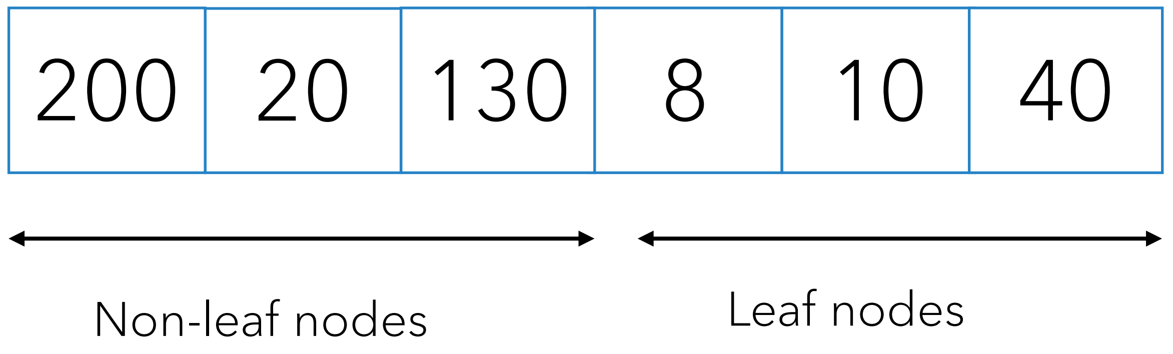
8	10	20	40	130	200

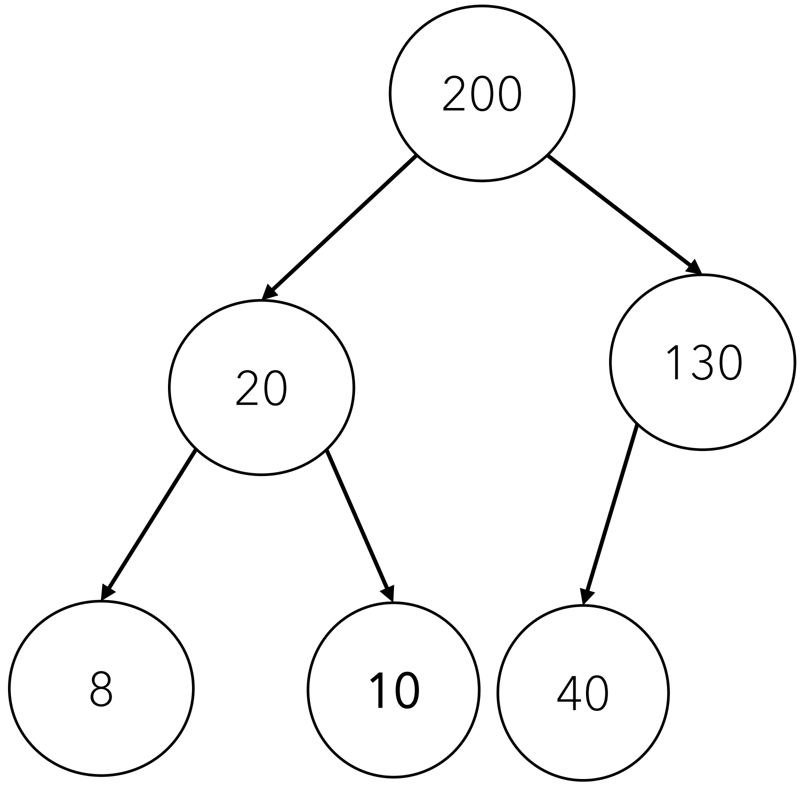
Build Max Heap

- Verify whether the property is satisfied for all nodes in the tree
- Top down vs Bottom up
- Bottom up heap construction
    - Verify the heap property starting from the last non-leaf node in a bottom up fashion
    - Apply heapify whenever the property is violated

Pause & Think

- If there is an array with n nodes organized as a binary tree , how many non leaf nodes will be there in the tree?





Function Build-Max-Heap(i)

```
Heap_size = n  # all nodes are part of the heap
for i  in range(n/2 to 0)
        Max-Heapify(A, i)
```

Summary

- Discussed on heap
- Learnt the procedure for transforming an array into heap

Thank You

Happy Learning

Success is always inevitable with Hard Work and Perseverance