COL 106 Lecture 11

Topic: Binary Heaps

Recap:

- Almost-Complete Binary Trees
- Array / List representation of Almost-Complete Binary Trees

Today:

- Heaps
- Priority Queues – Efficient Enqueue and Extract Min

Claim: Let $k_0, k_1, \ldots, k_{n-1}$ be the level-order traversal of an almost-complete binary tree. Then the parent of k_m is $k_{\lfloor \frac{m-1}{2} \rfloor}$, its left child is k_{2m+1}, and its right child is k_{2m+2}.

Proof: Number nodes in each level left to right, starting from 0. If k_m is the j^{th} node in the i^{th} level, then

$$m = \underbrace{1}_{\text{level 0}} + \underbrace{2}_{\text{level 1}} + \cdots + \underbrace{2^{i-1}}_{\text{level } i-1} + \underbrace{j}_{\text{level } i} + \underbrace{1}_{\substack{\text{because level order} \\ \text{numbering} \\ \text{starts from 0.}}}$$

$$= 2^i - 1 + j$$

j indexing starts with 0

The $\begin{Bmatrix} \text{left} \\ \text{right} \end{Bmatrix}$ child of the j^{th} node in level i

is the $\begin{Bmatrix} 2j \\ 2j+1 \end{Bmatrix}^{th}$ node in level $i+1$.

∴ The children of k_m are $k_{2^{i+1}-1+2j}$ and $k_{2^{i+1}-1+2j+1}$

Recall $m = 2^i - 1 + j$

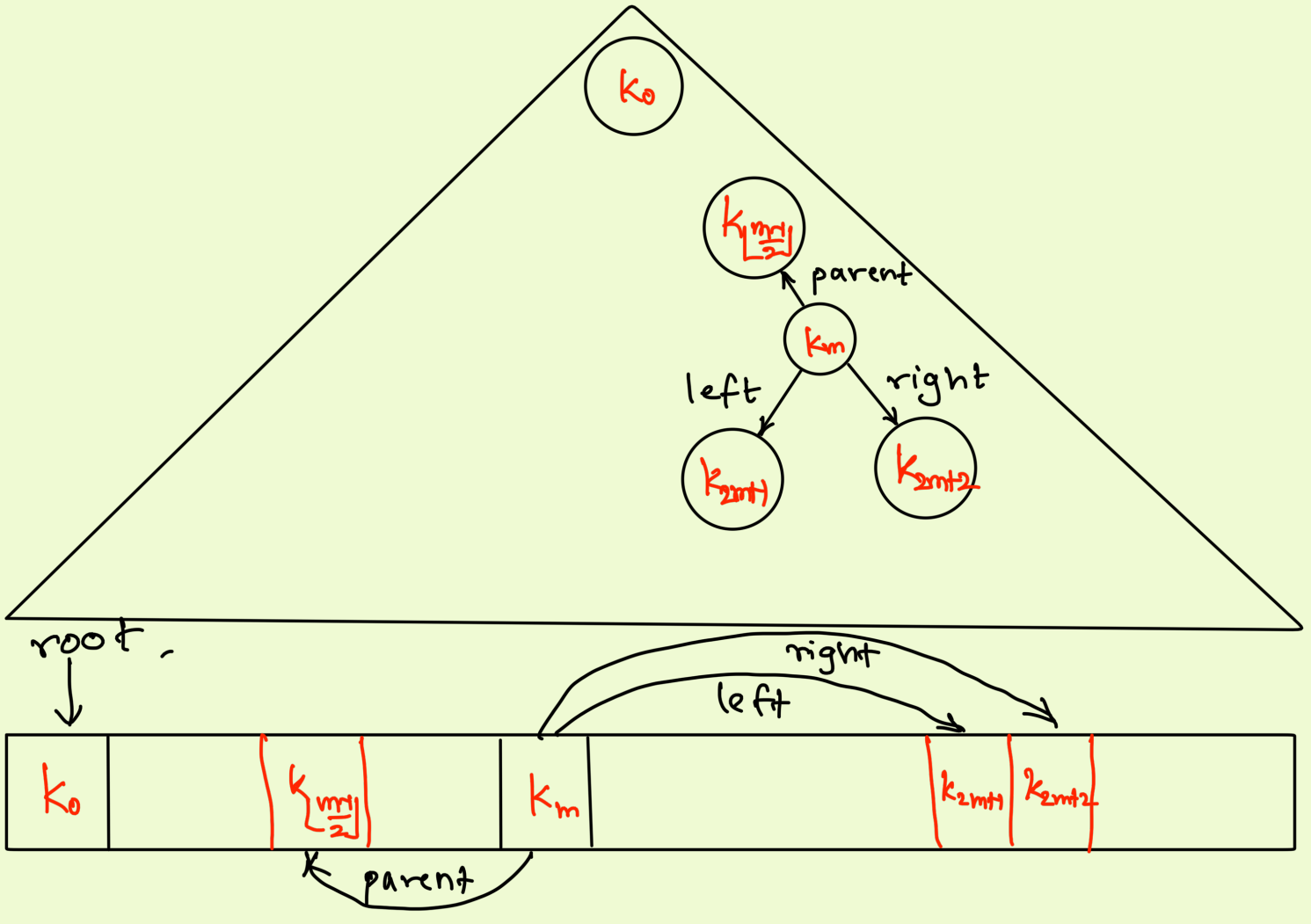
∴ Left child : $k_{2^{i+1}-1+2j} = k_{2m+1}$

Right child : $k_{2^{i+1}+2j} = k_{2m+2}$.

∴ Parent of $k_m = k_{\lfloor \frac{m-1}{2} \rfloor}$

Array / List representation of Almost-Complete Binary Trees

Idea: Represent an almost-complete binary tree by its level-order traversal



Binary Heaps

Definition: Let $(S, \leq)$ be an ordered set. A binary tree T with keys from S is called a binary min-heap if it satisfies the following two properties.

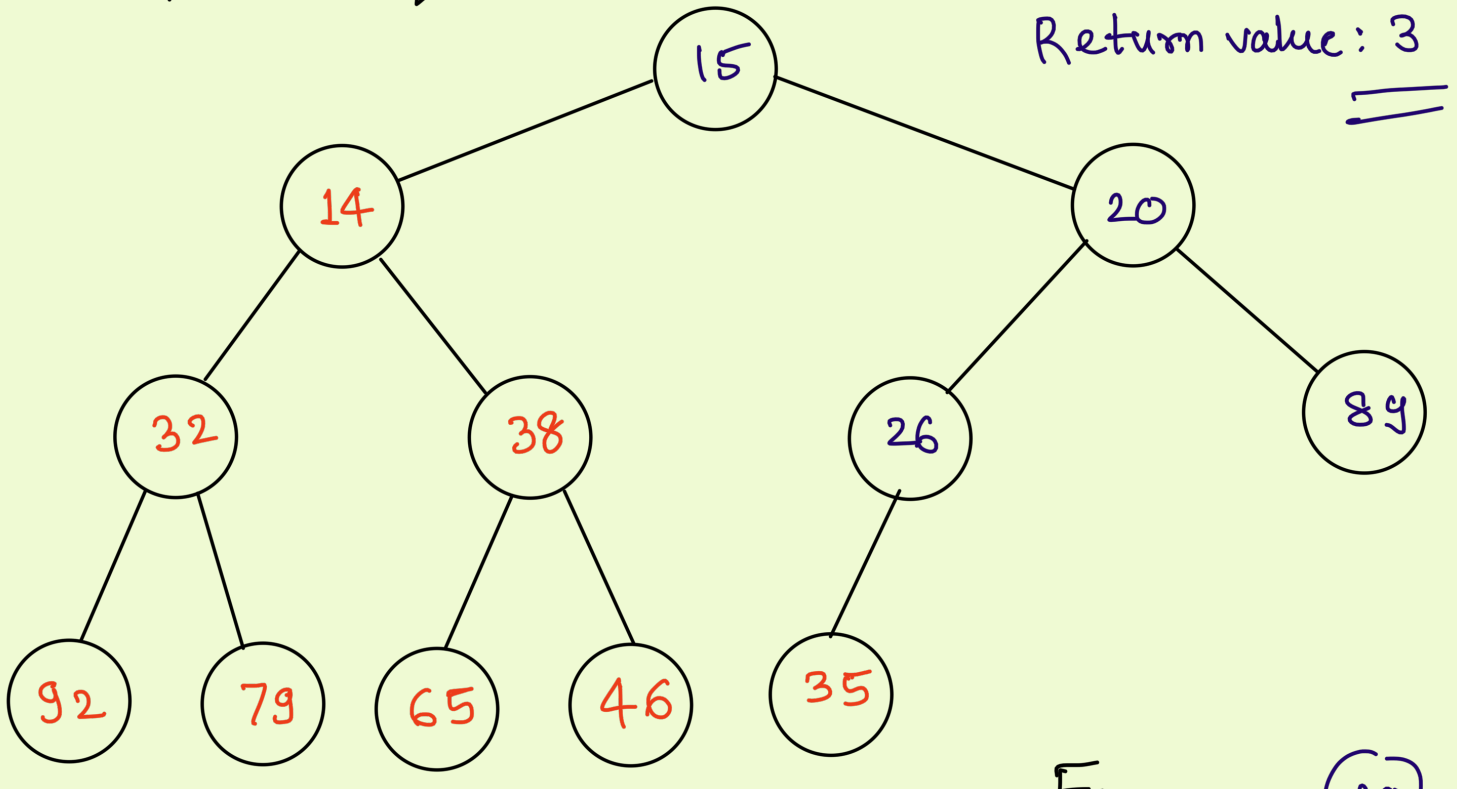
1. [Structure] T is an almost-complete binary tree.
2. [Order] For each node $u \neq$ root:

$$u.\text{parent}.\text{key} \leq u.\text{key}.$$

(Binary max-heaps are defined analogously.)

Priority Queue implementation: Maintain elements in a min-heap.

Example: Enqueue and Extract Min



Return value: 3

Enqueue (20)

Enqueue (x):

1. Attach a new node with key k at the leftmost available place in the bottommost layer
   (Create a new layer if needed)
   (ie. append(x) in the array / list representation.)
2. While $x <$ key of x's parent
   Exchange x with the key of its parent.

Time Complexity : $O(h) = O(\log n)$

Why is this correct? → Next class

Extract Min ()

1. $w \leftarrow$ key of root.
2. Let u be the rightmost node in the bottommost layer. Let x be the key of u.
   Disconnect u from its parent.
   (pop() in the array / list representation)
3. key of root $\leftarrow x$
4. While x has a child with key $< x$
   - Exchange x with the min of its children's keys.
5. Return w.

Time Complexity : $O(h) = O(\log n)$ Correctness → next class