DIT181: Data Structures and Algorithms

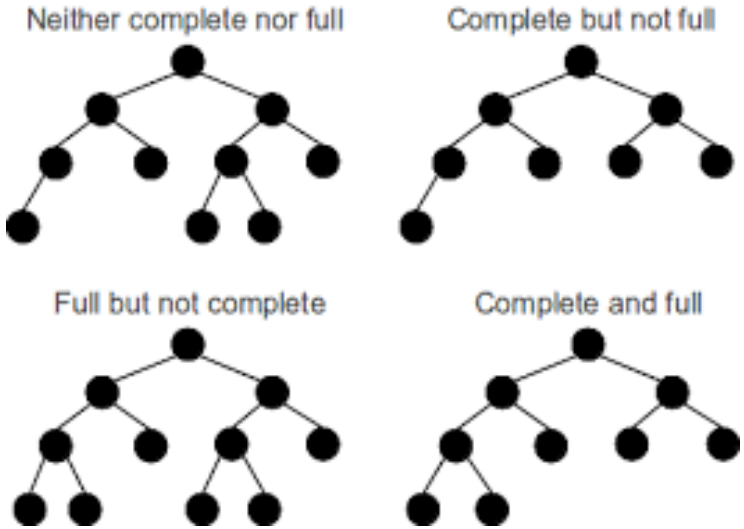
A Priority Queue: The Binary Heap

Gül Calikli
Email: calikli@chalmers.se

Kahoot Questions

Kahoot link: https://create.kahoot.it/l/#user/786e1f87-d4ef-4ed0-99c6-cbf9a7284c22/kahoots/created

A "complete binary tree" is a tree that is completely filled with possible exception of the bottom level, which is filled from left to right and has no missing nodes.



What is a Priority Queue?

- The “priority queue” is a data structure supports the access and deletion of the **minimum** item with methods findMin()and deleteMin()

What is a Priority Queue?

- The “priority queue” is a data structure supports the access and deletion of the **minimum** item with methods findMin()and deleteMin()
- How can we implement priority queues?

What is a Priority Queue?

- The “priority queue” is a data structure supports the access and deletion of the **minimum** item with methods findMin()and deleteMin()
- How can we implement priority queues?
- How about using **arrays** or **linked lists**?

What is a Priority Queue?

- The “priority queue” is a data structure supports the access and deletion of the **minimum** item with methods findMin()and deleteMin()
- How can we implement priority queues?
- How about using **arrays** or **linked lists**?
- Question: What would be the big-O complexity of finding and deleting the minimum?

What is a Priority Queue?

- The “priority queue” is a data structure supports the access and deletion of the **minimum** item with methods findMin()and deleteMin()
- How can we implement priority queues?
- How about using **arrays** or **linked lists**?
- Question: What would be the big-O complexity of finding and deleting the minimum?
- Answer: O(N)

What is a Priority Queue?

- The “priority queue” is a data structure supports the access and deletion of the **minimum** item with methods findMin()and deleteMin()
- How can we implement priority queues?
- How about using **sorted linked lists**?
- Question: What would be the big-O complexity of finding and deleting the minimum?

What is a Priority Queue?

- The “priority queue” is a data structure supports the access and deletion of the **minimum** item with methods findMin()and deleteMin()
- How can we implement priority queues?
- How about using **a binary search tree**?
- Question: What would be the big-O complexity of finding and deleting the minimum?

What is a Priority Queue?

- The “priority queue” is a data structure supports the access and deletion of the **minimum** item with methods findMin()and deleteMin()
- How can we implement priority queues?
- How about using **a binary search tree**?
- Question: What would be the worst big-O complexity of finding and deleting the minimum?
- Answer: O(N)

What is a Priority Queue?

- The “priority queue” is a data structure supports the access and deletion of the **minimum** item with methods findMin()and deleteMin()
- How can we implement priority queues?
- How about using **a balanced binary search tree**?
- Question: What would be the big-O complexity of finding and deleting the minimum?

What is a Priority Queue?

- The “priority queue” is a data structure supports the access and deletion of the **minimum** item with methods findMin()and deleteMin()
- How can we implement priority queues?
- How about using **a balanced binary search tree**?
- Question: What would be the big-O complexity of finding and deleting the minimum?
- Answer: O(logN)

What is a Priority Queue?

- The “priority queue” has properties that is a compromise between a queue and a balanced binary search tree.
- The “priority queue” can be implemented by a binary heap, which realizes this compromise.

Binary Heap

- Using a “binary heap”, a priority queue can be implemented as follows:
  - Can be implemented by using a simple array → like the queue
  - Supports insert and deleteMin operations in O(logN) worst time complexity → a compromise between binary search tree and the queue
  - Supports insert in constant O(1) average time → like the queue
  - Supports findMin in constant O(1) worst case time → like the queue

Binary Heap

- A binary heap has two properties:
  - A structure property, and
  - A (heap) order property

Binary Heap

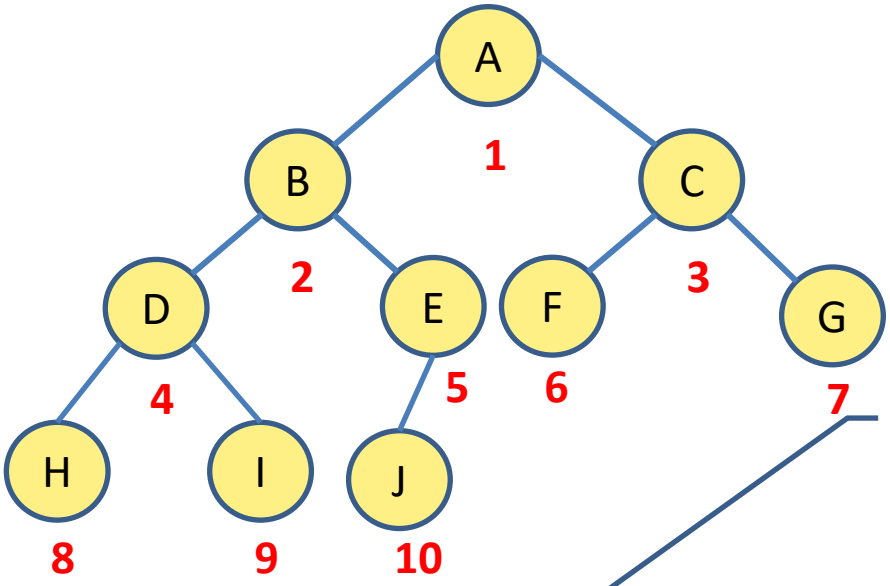
- A binary heap has two properties:
  - A structure property, and
  - A (heap) order property

Binary Heap: Structure Property

- A ”complete binary tree” has a set of powerful properties:
- The following holds for the height H (longest path) of a binary tree: $2^H \leq H \leq 2^{H+1}$
- Hence, a complete binary tree has at most $\lfloor logN \rfloor$.

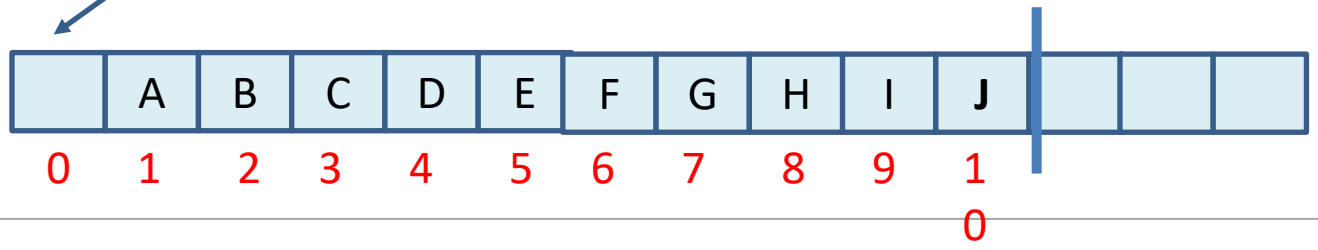
Binary Heap: Structure Property

- A ”complete binary tree” does not need left and right links.
- We can represent a “complete binary tree” by storing its elements in level order traversal in an array.



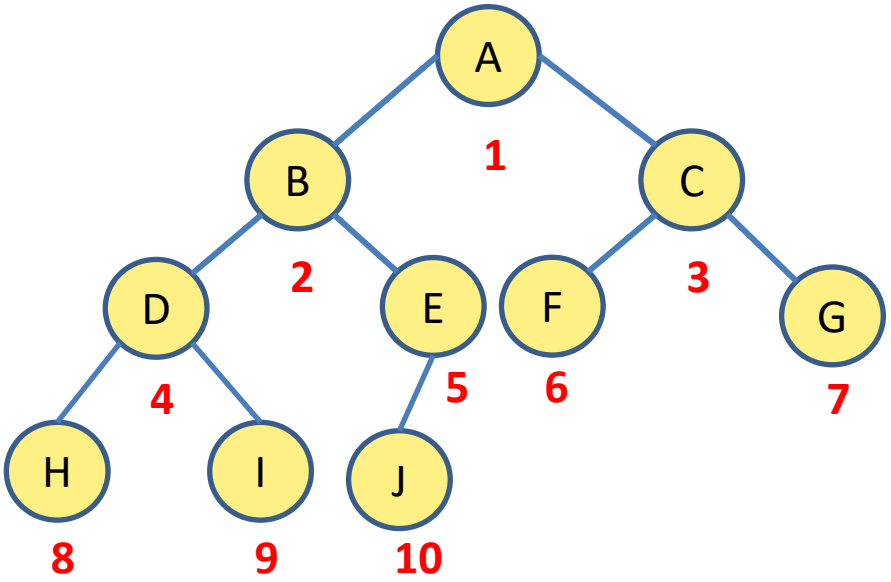
A complete binary tree and its array representation.

Position 0 is often left vacant



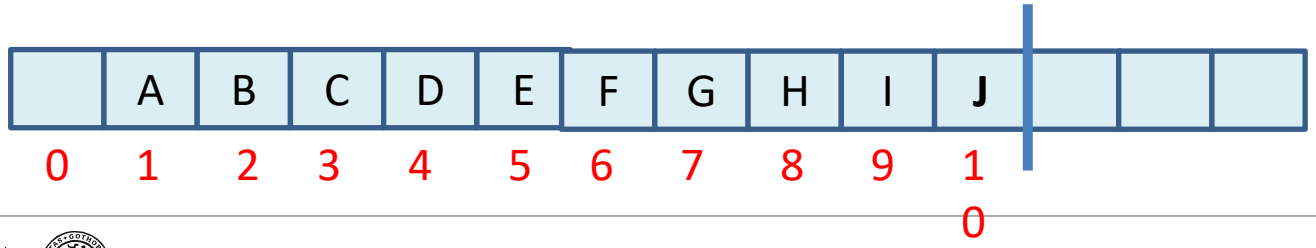
Binary Heap: Structure Property

- A ”complete binary tree” does not need left and right links.
- We can represent a “complete binary tree” by storing its elements in level order traversal in an array.



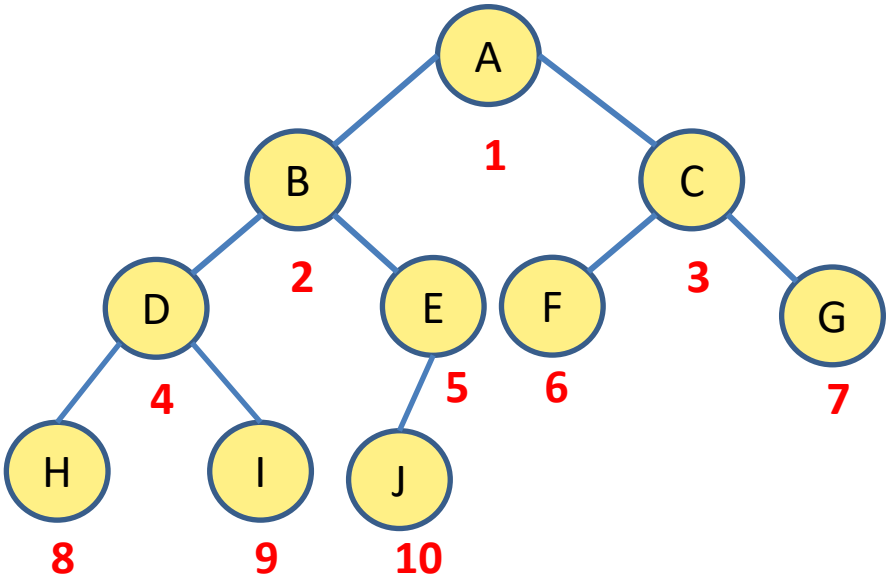
A complete binary tree and its array representation.

For any element in position **i**, left child is at position **2i** and right element is at position **2i+1**, if they exist. We need to check the actual size of the tree.



Binary Heap: Structure Property

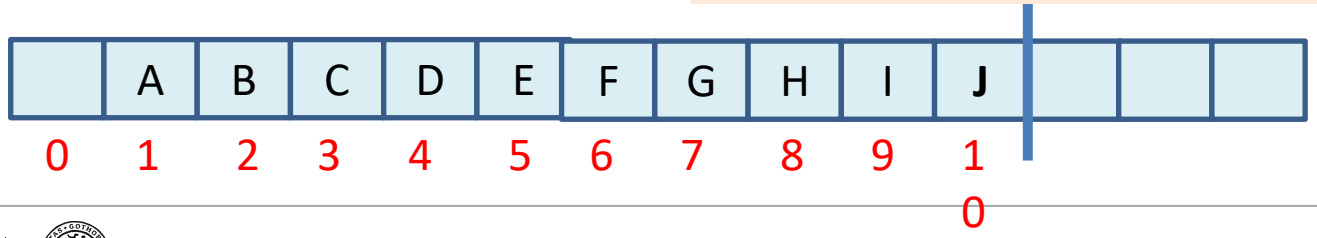
- A ”complete binary tree” does not need left and right links.
- We can represent a “complete binary tree” by storing its elements in level order traversal in an array.



A complete binary tree and its array representation.

For any element in position **i**, left child is at position **2i** and right element is at position **2i+1**, if they exist. We need to check the actual size of the tree.

The parent of an element at position I is at $\lfloor i/2 \rfloor$



Binary Heap: Structure Property

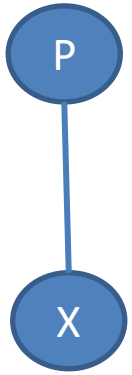
- **In-Class Exercise 9.1:** Suppose the binary heap is stored with the root at position r. Give the formulas for the locations of the children and parent of the node in position i.

Binary Heap

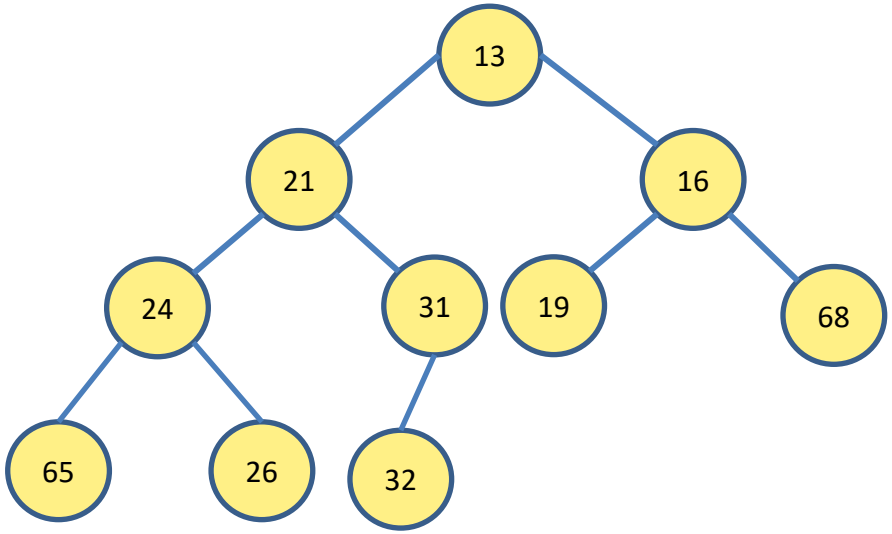
- A binary heap has two properties:
  - A structure property, and
  - A (heap) order property

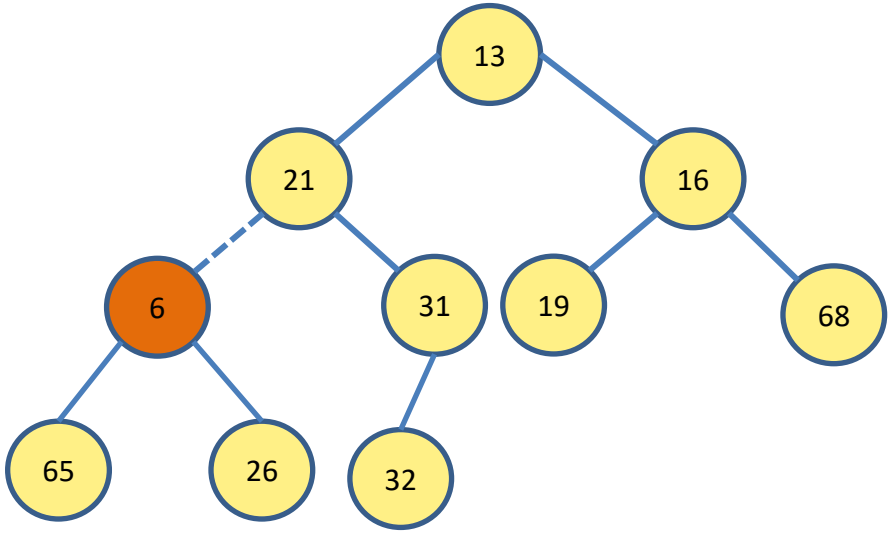
Binary Heap: Heap order property

- Heap order property: In a heap, for every node X with parent P, the key in P is smaller than or equal to the key in X.



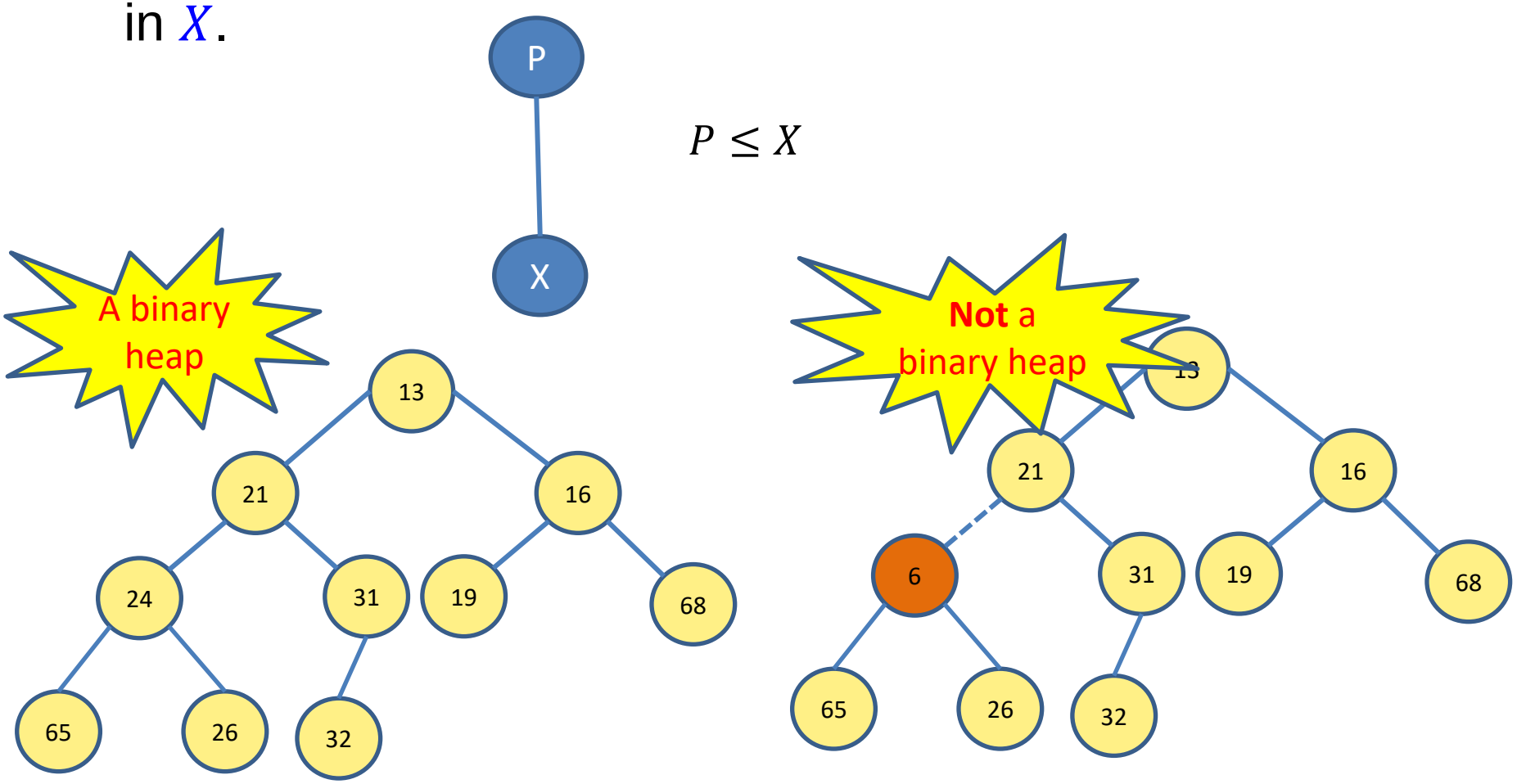
$$P \leq X$$



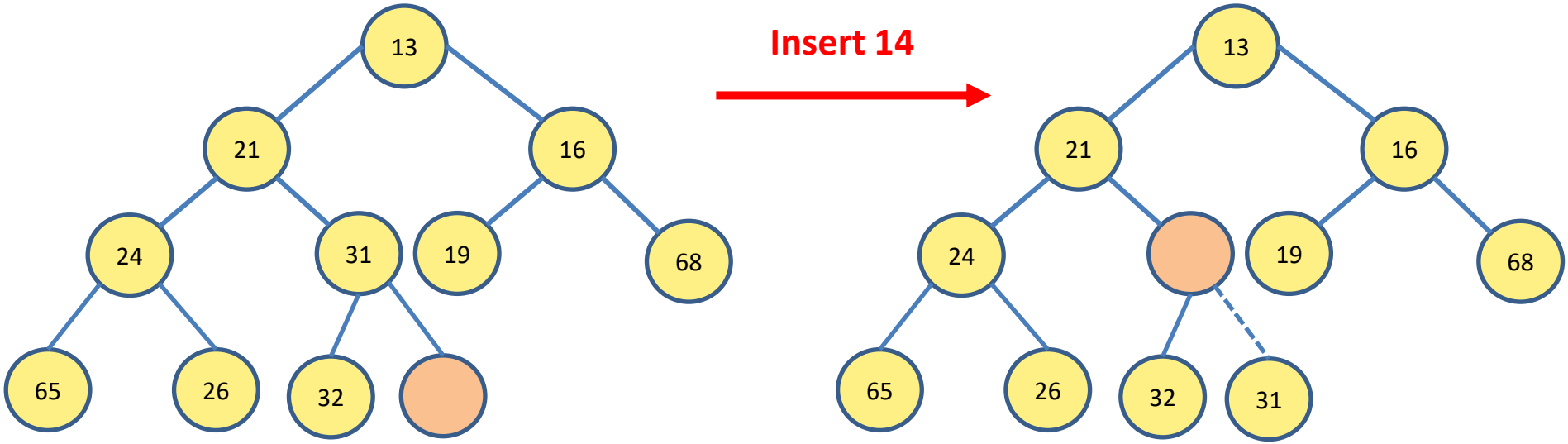


Binary Heap: Heap order property

- Heap order property: In a heap, for every node X with parent P, the key in P is smaller than or equal to the key in X.



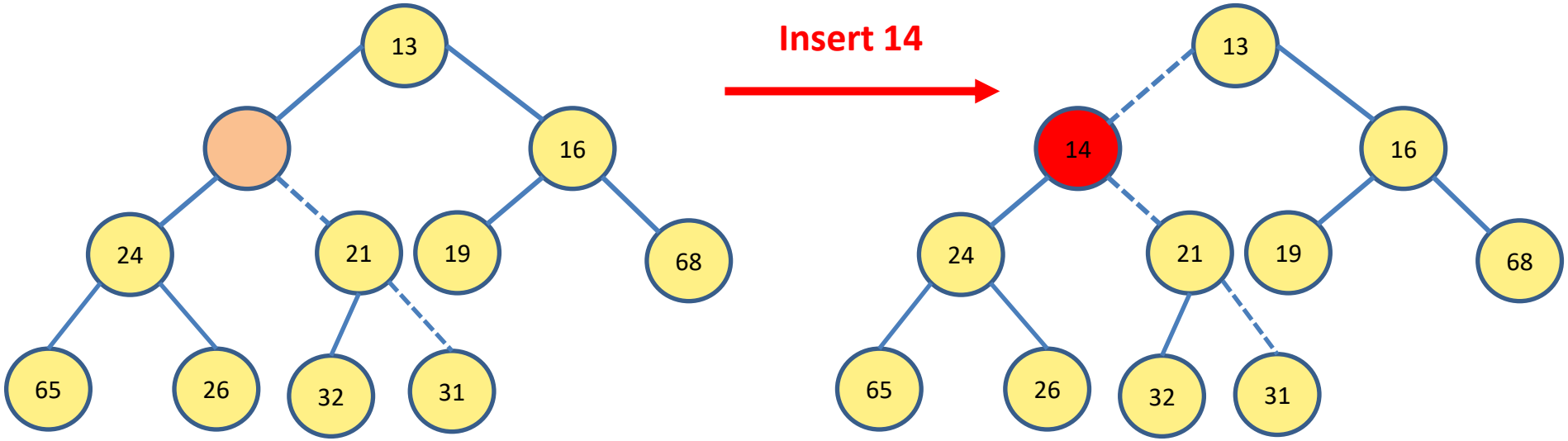
Binary Heap Basic Operations: Insertion



Create a hole

Bubble up the hole

Binary Heap Basic Operations: Insertion



Create a hole

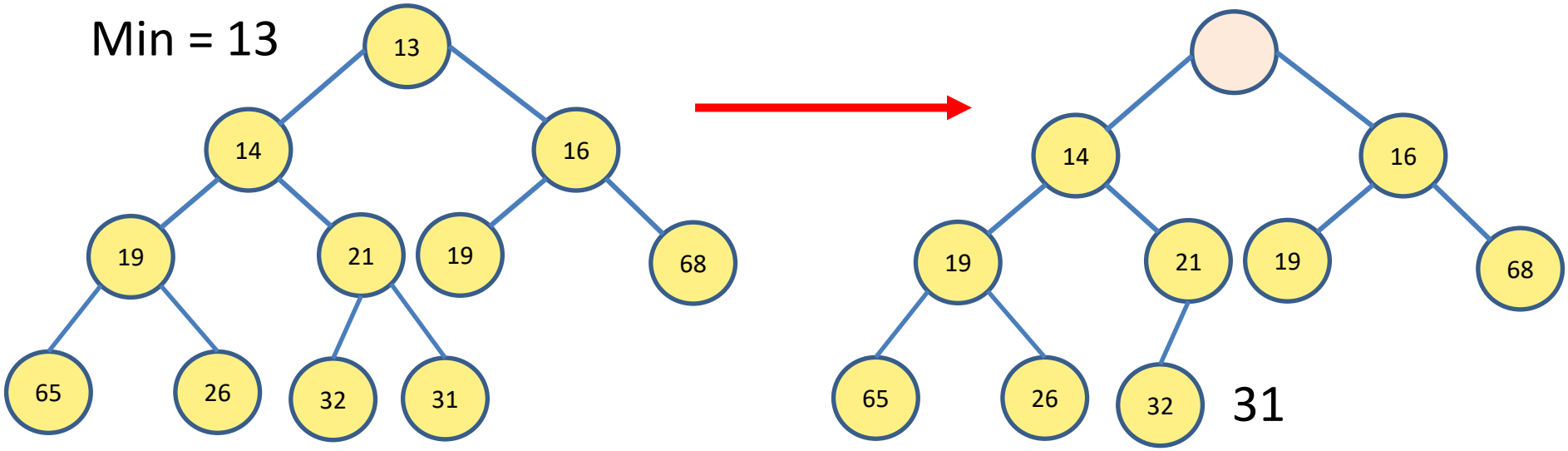
Bubble up the hole

This is called Percolate Up

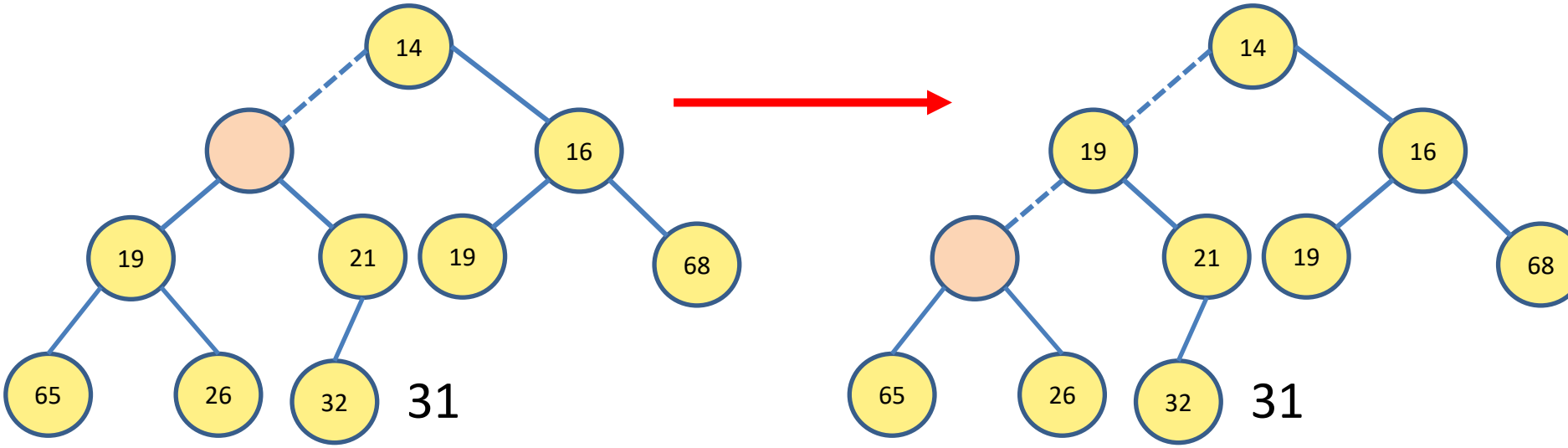
Binary Heap Basic operations: insert

- **In-Class Exercise 9.2:** Implement the insert method for the binary heap tree (skeleton code is given on the exercise sheets).

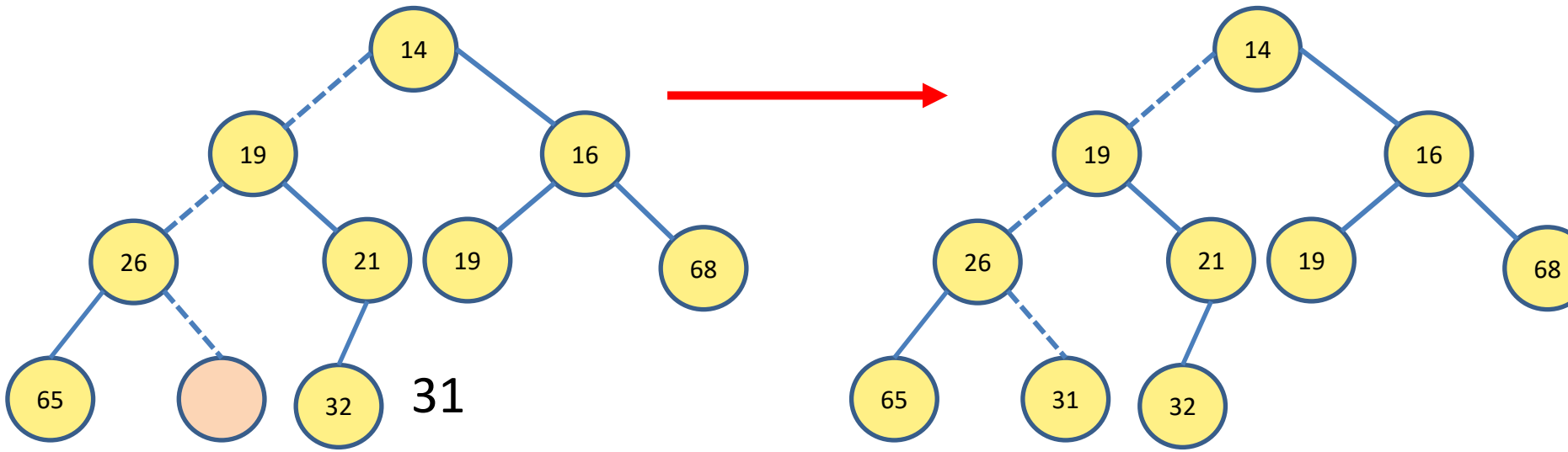
Binary Heap Basic Operations: deleteMin



Binary Heap Basic Operations: deleteMin



Binary Heap Basic Operations: deleteMin

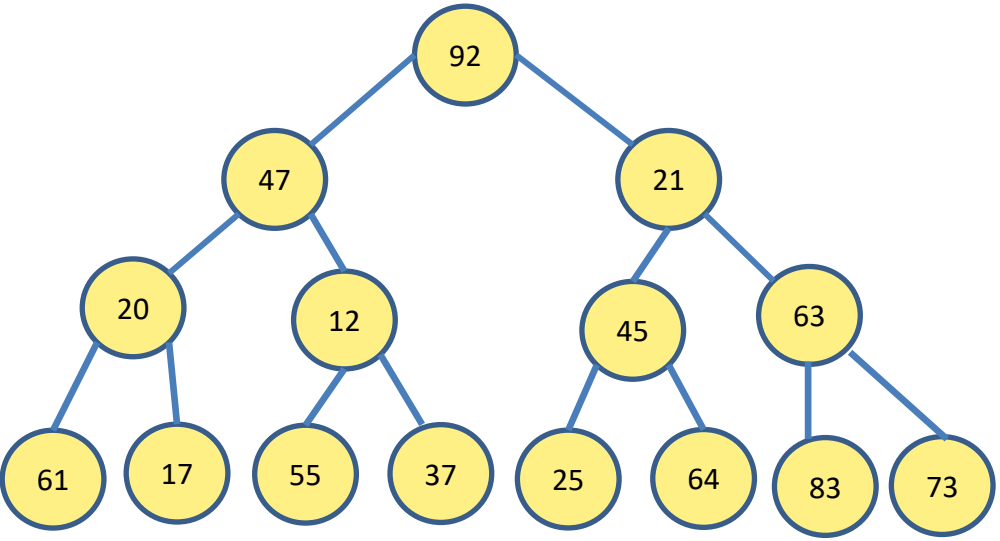


This is called Percolate Down

Binary Heap Basic operations: insert

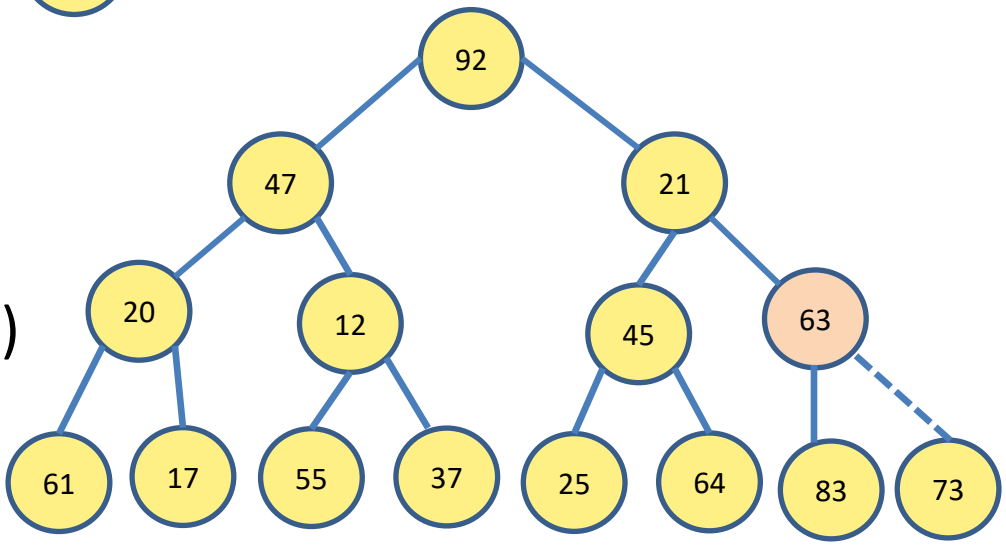
- **In-Class Exercise 9.3:** Implement the percolateDown() method that is called by the remove() method method for the binary heap tree (skeleton code is given on the exercise sheets).

Binary Heap Basic operations: buildHeap()

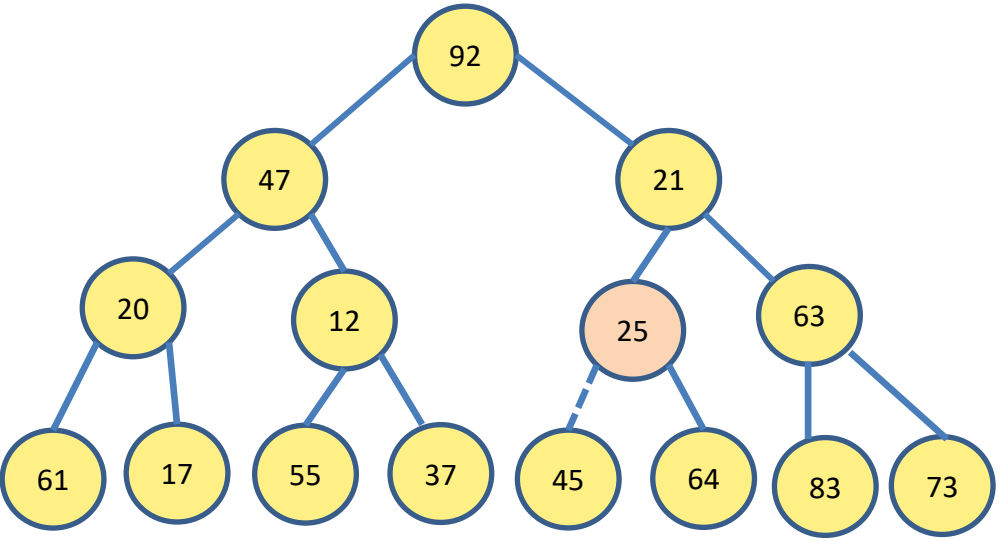


Initial heap

After percolatedown(7)

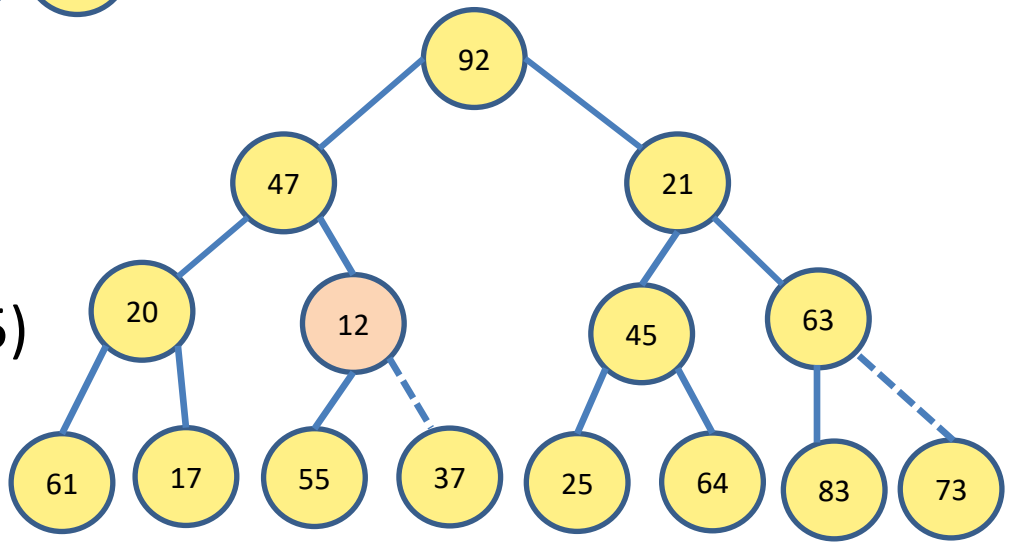


Binary Heap Basic operations: buildHeap()

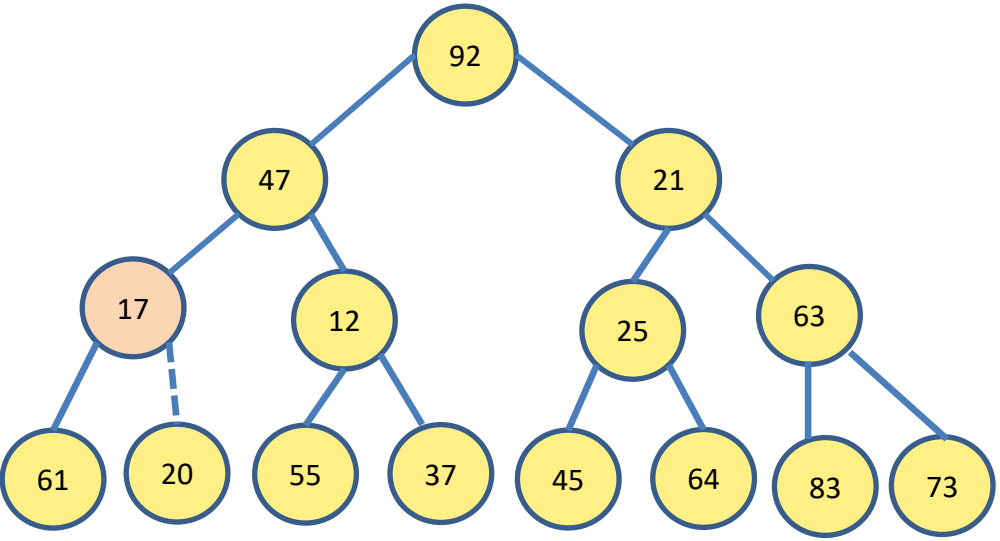


After percolateDown(6)

After percolateDown(5)

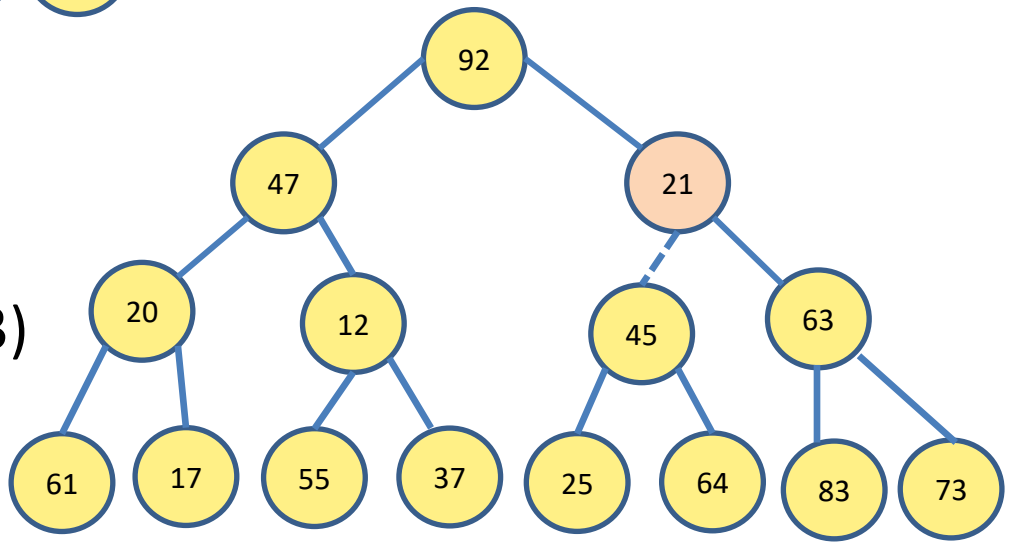


Binary Heap Basic operations: buildHeap()

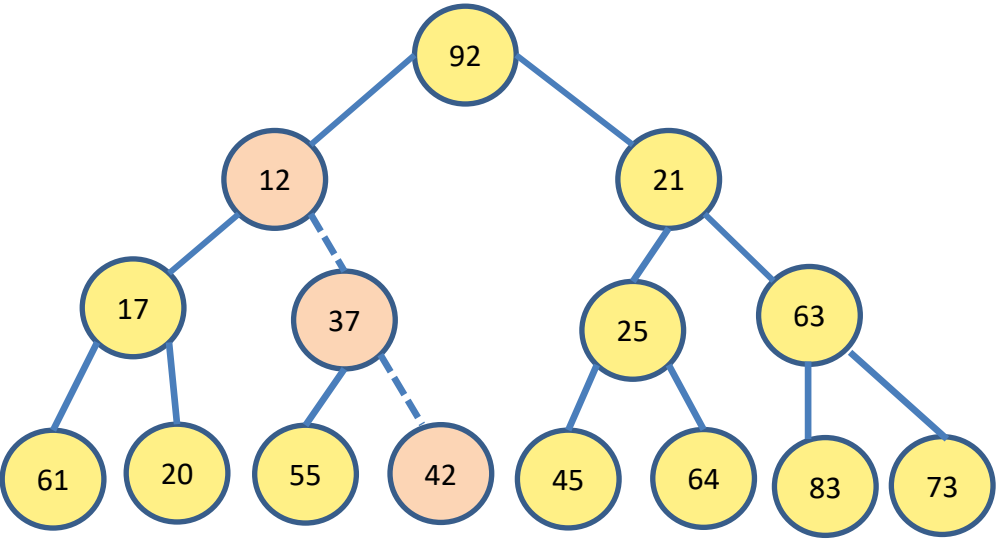


After percolatedown(4)

After percolateDown(3)

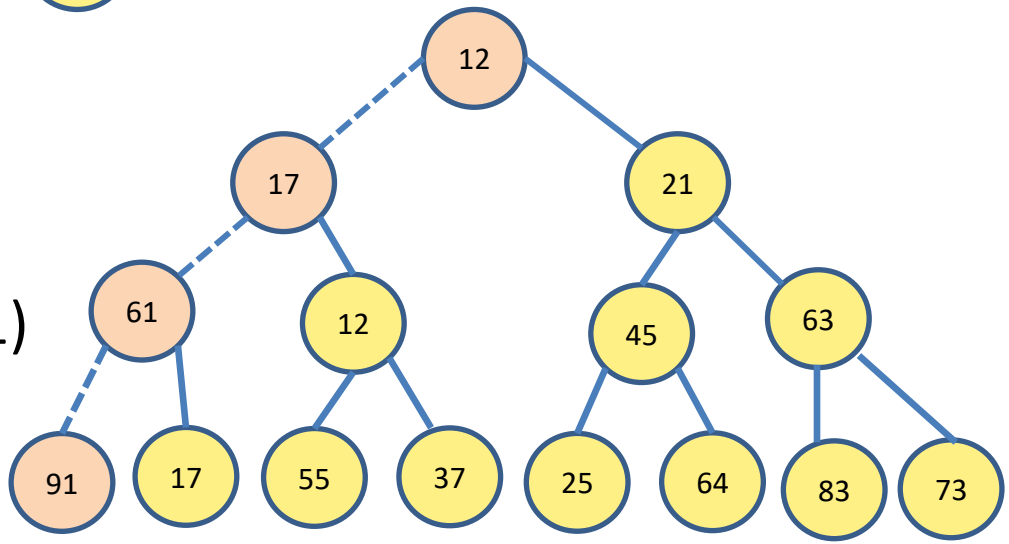


Binary Heap Basic operations: buildHeap()



After percolateDown(2)

After percolateDown(1)



Binary Heap Basic operations: insert

- **In-Class Exercise 9.4:** Implement the buildHeap() method (Tip: use percolateDown() method) (skeleton code is given on the exercise sheets).