Algoritme dan Struktur Data

Heap Tree

Putra Pandu Adikara
Fakultas Ilmu Komputer
Universitas Brawijaya

Heaps

- A heap is a certain kind of complete binary tree.

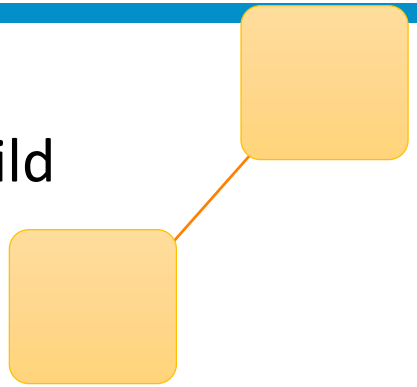
Heaps

- A heap is a certain kind of complete binary tree.

When a complete binary tree is built, its first node must be the root.

Heaps

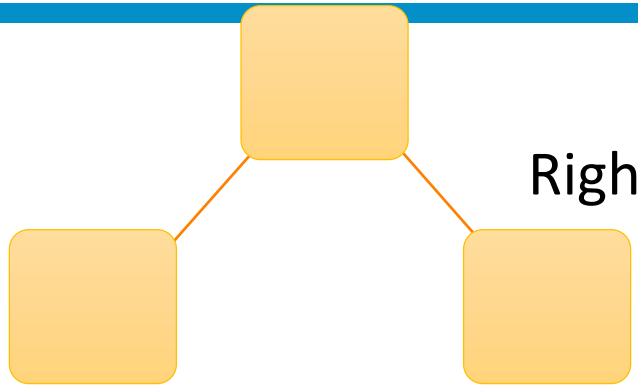
- Complete binary tree.



The second node is always the left child of the root.

Heaps

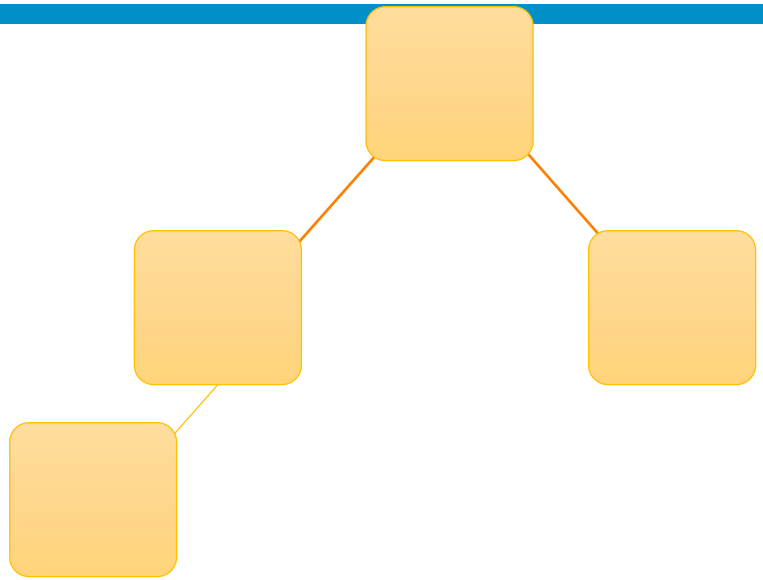
- Complete binary tree.



The third node is always the right child of the root.

Heaps

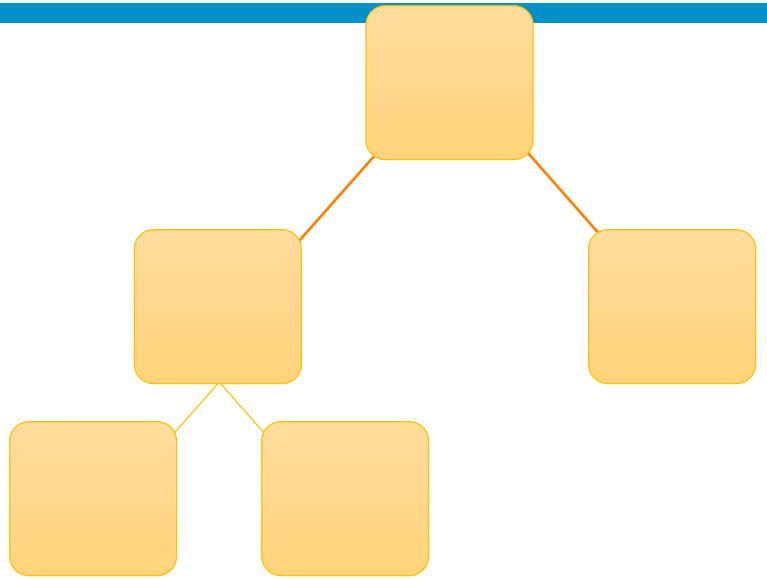
- Complete binary tree.



The next nodes always fill the next level from left-to-right.

Heaps

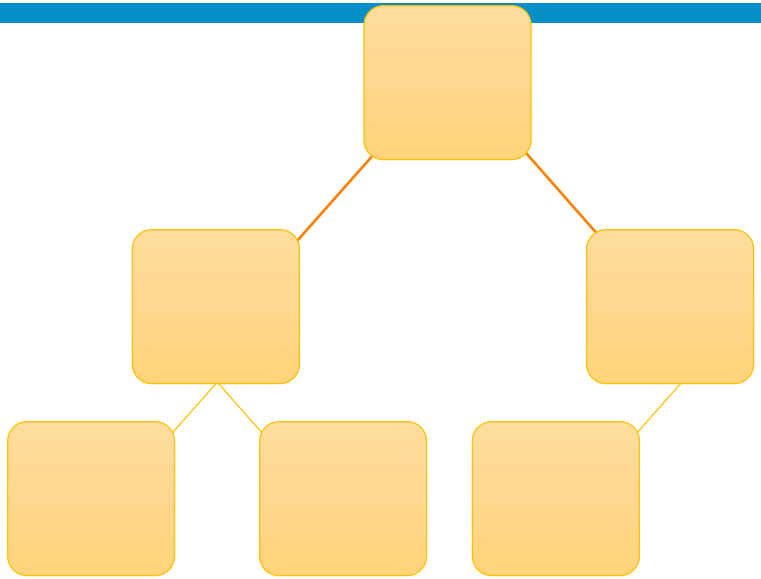
- Complete binary tree.



The next nodes always fill the next level from left-to-right.

Heaps

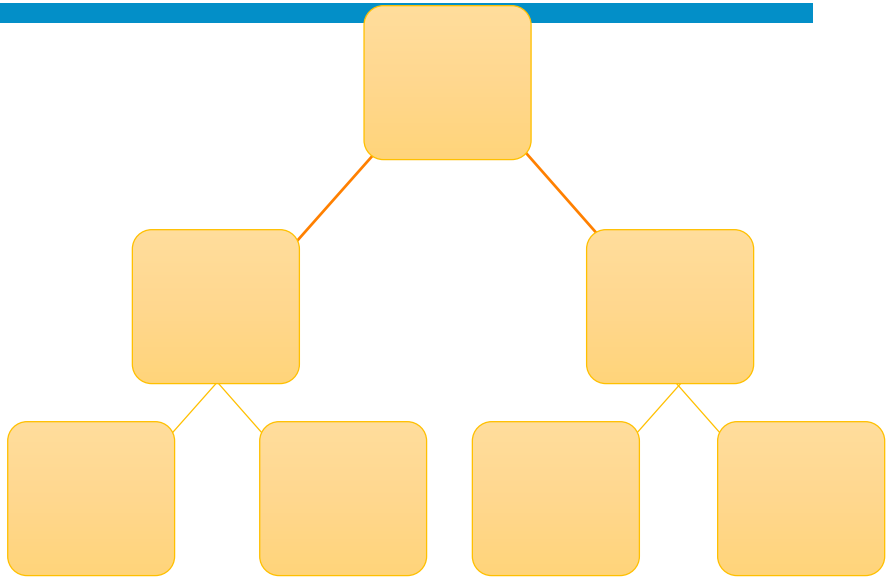
- Complete binary tree.



The next nodes always fill the next level from left-to-right.

Heaps

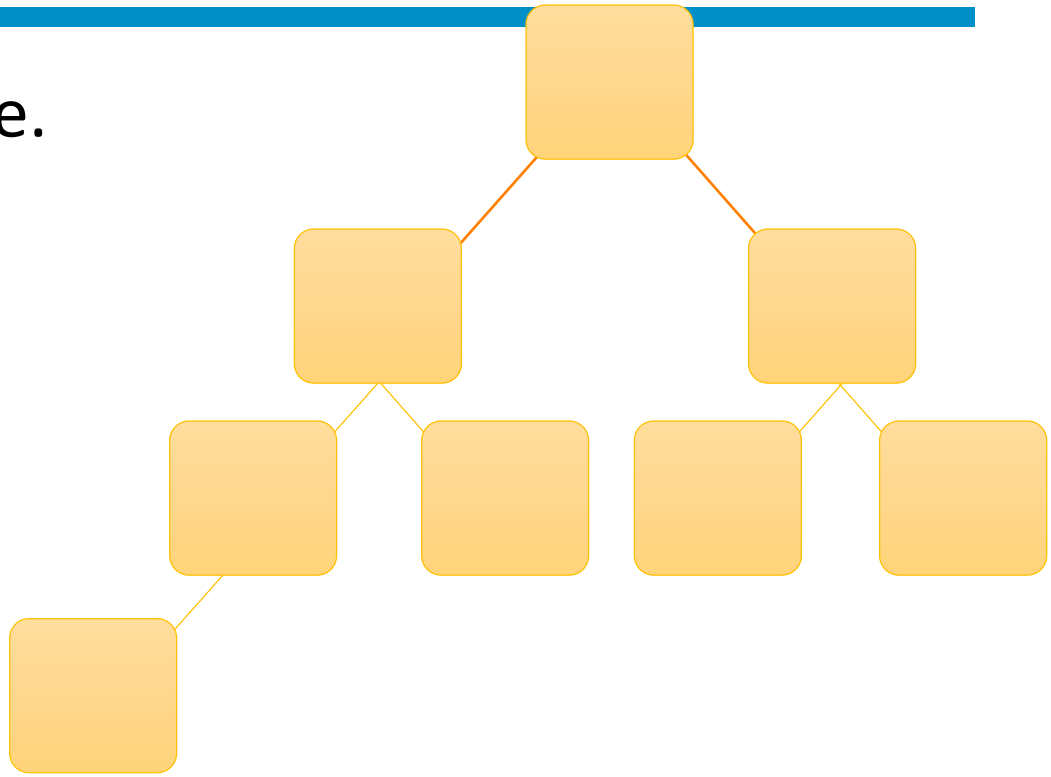
- Complete binary tree.



The next nodes always fill the next level from left-to-right.

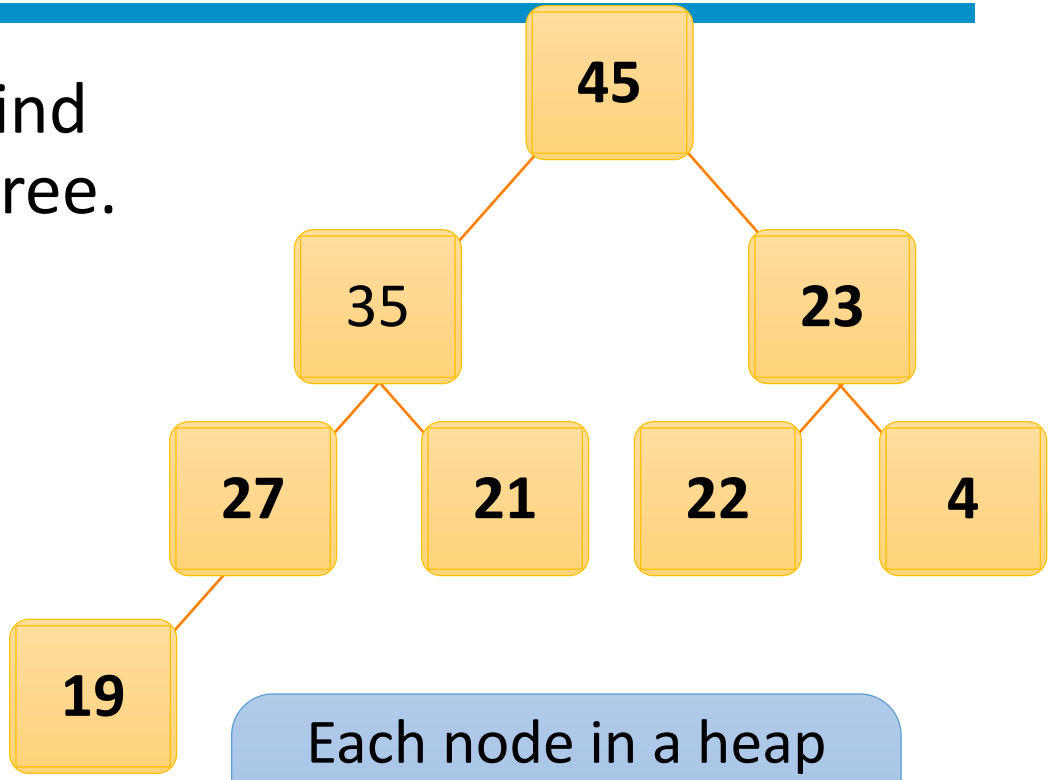
Heaps

- Complete binary tree.



Heaps

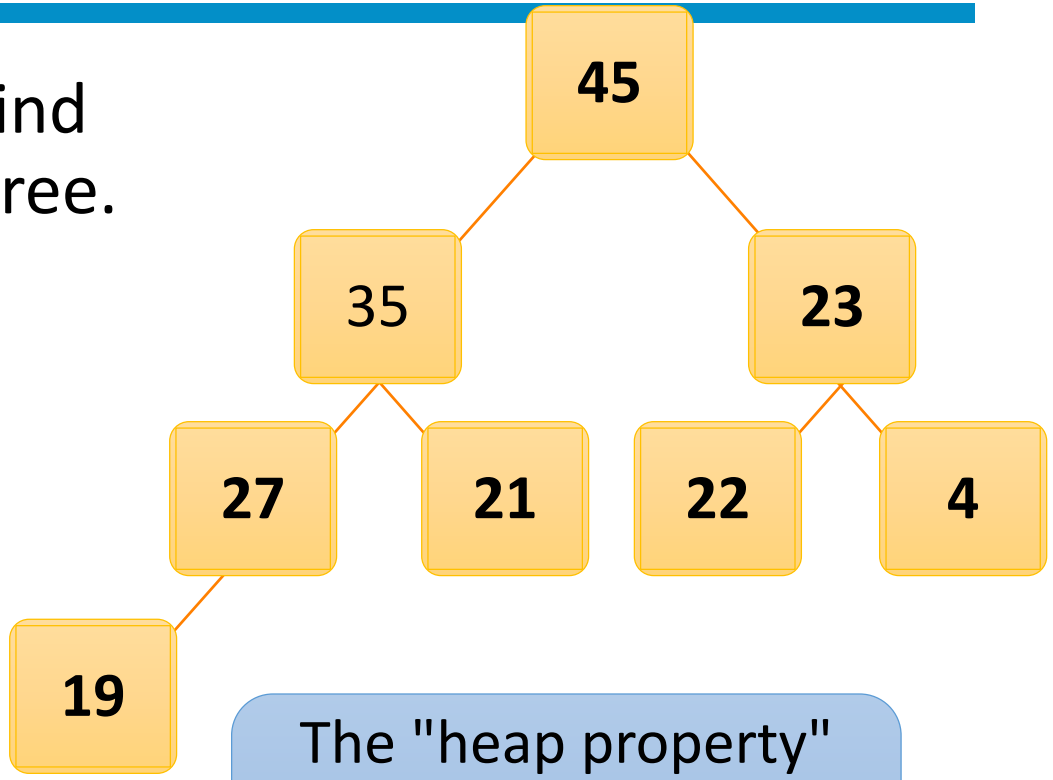
- A heap is a certain kind of complete binary tree.



Each node in a heap contains a key that can be compared to other nodes' keys.

Heaps

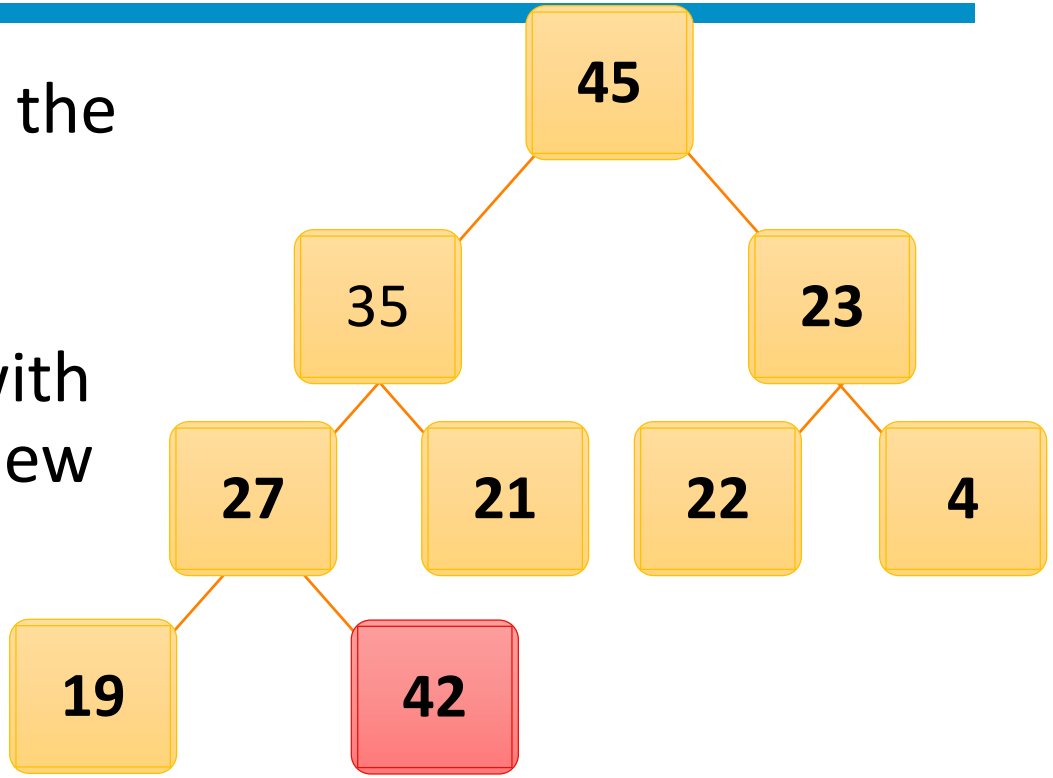
- A heap is a certain kind of complete binary tree.



The "heap property" requires that each node's key is >= the keys of its children

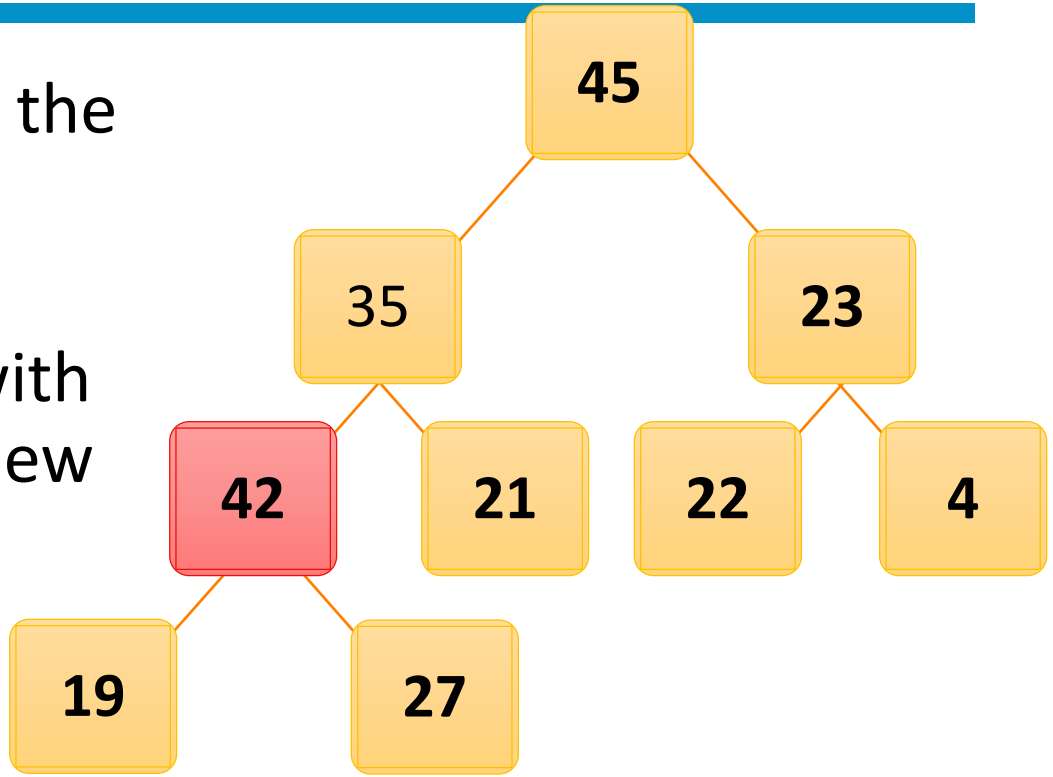
Adding a Node to a Heap

- Put the new node in the next available spot.
- Push the new node upward, swapping with its parent until the new node reaches an acceptable location.



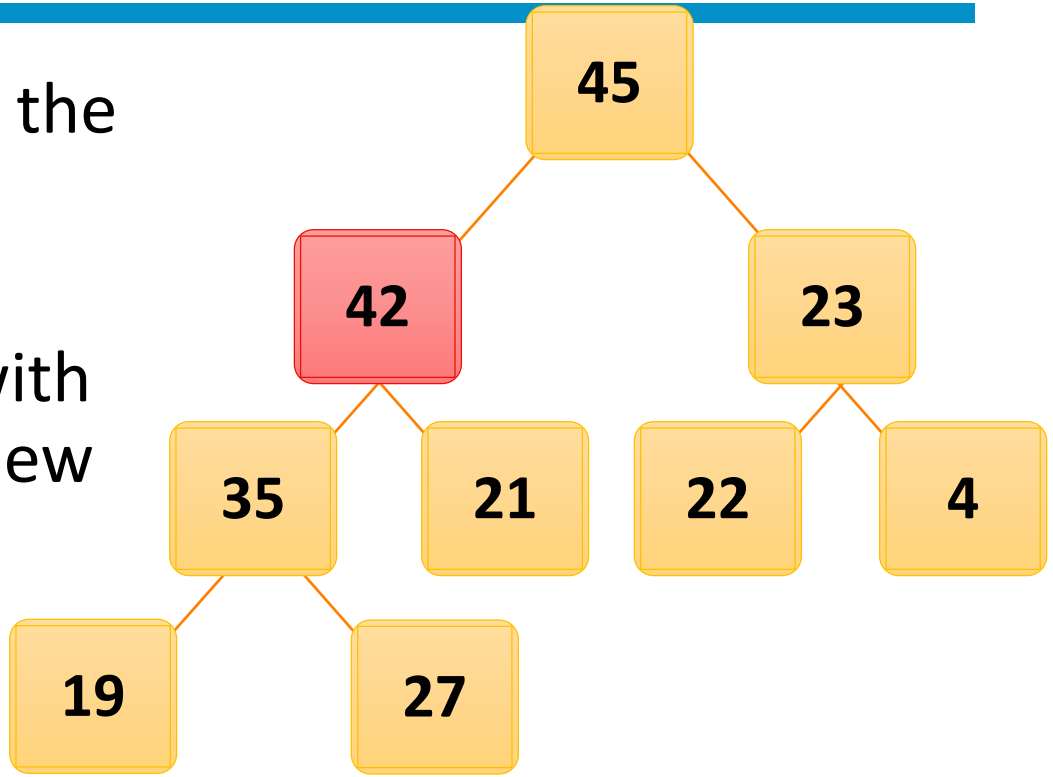
Adding a Node to a Heap

- Put the new node in the next available spot.
- Push the new node upward, swapping with its parent until the new node reaches an acceptable location.



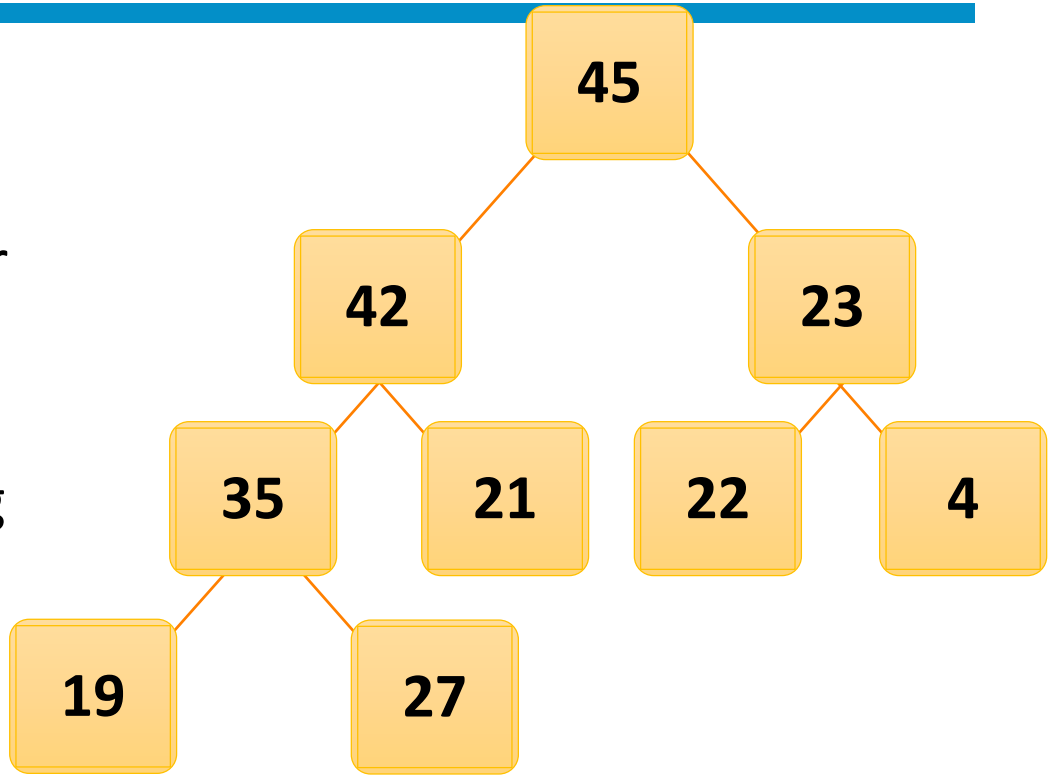
Adding a Node to a Heap

- Put the new node in the next available spot.
- Push the new node upward, swapping with its parent until the new node reaches an acceptable location.



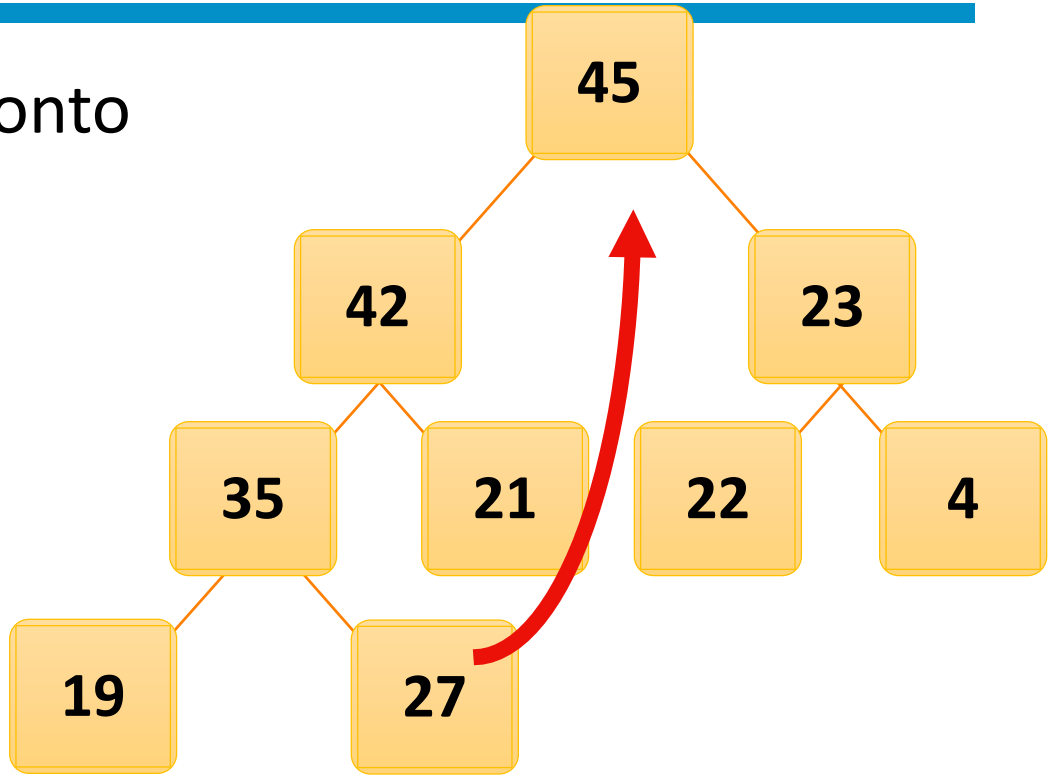
Adding a Node to a Heap

- The parent has a key that is >= new node, or
- The node reaches the root.
- The process of pushing the new node upward is called **reheapification upward**.



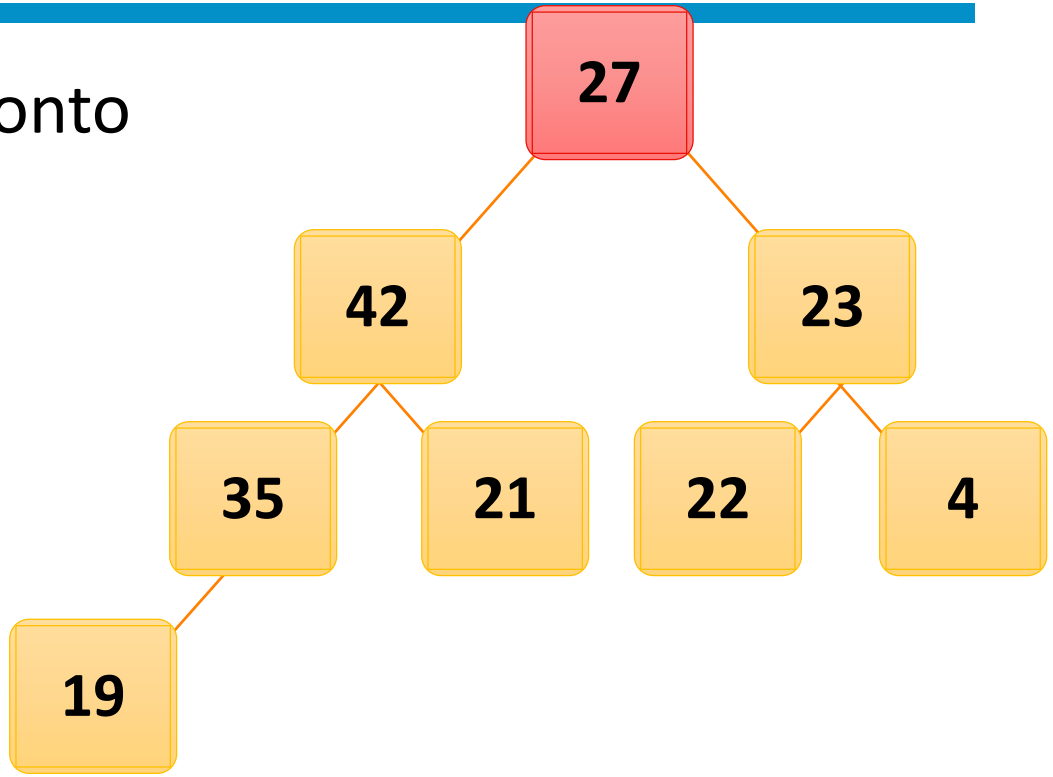
Removing the Top of a Heap

- Move the last node onto the root.



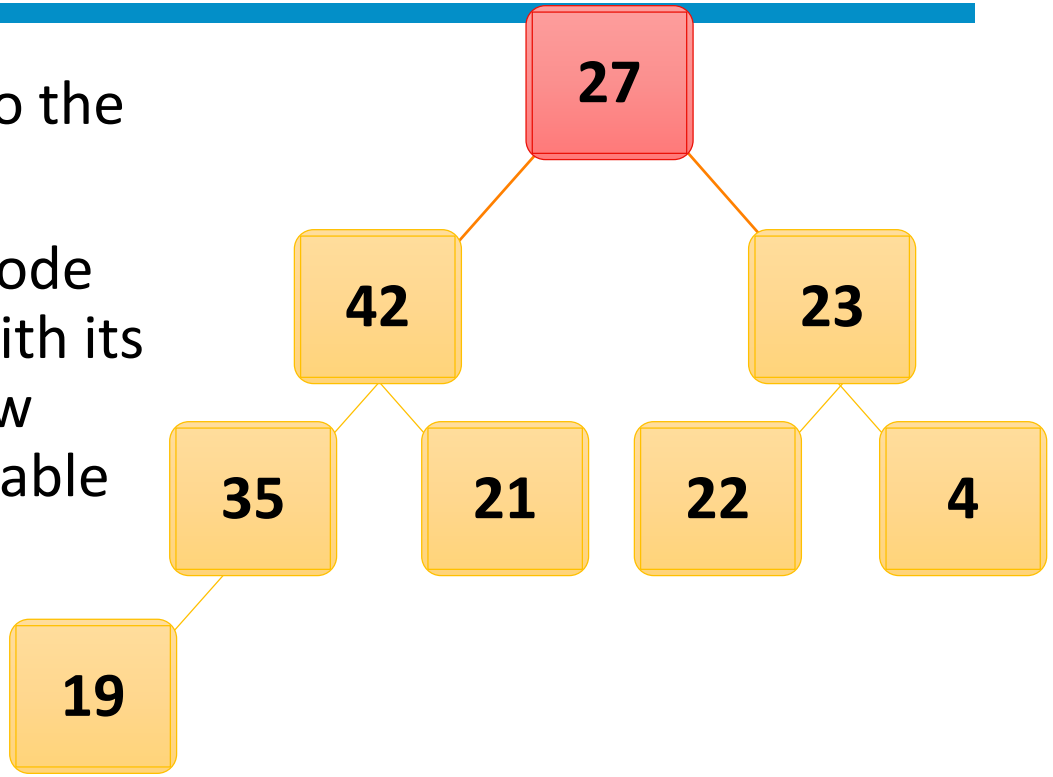
Removing the Top of a Heap

- Move the last node onto the root.



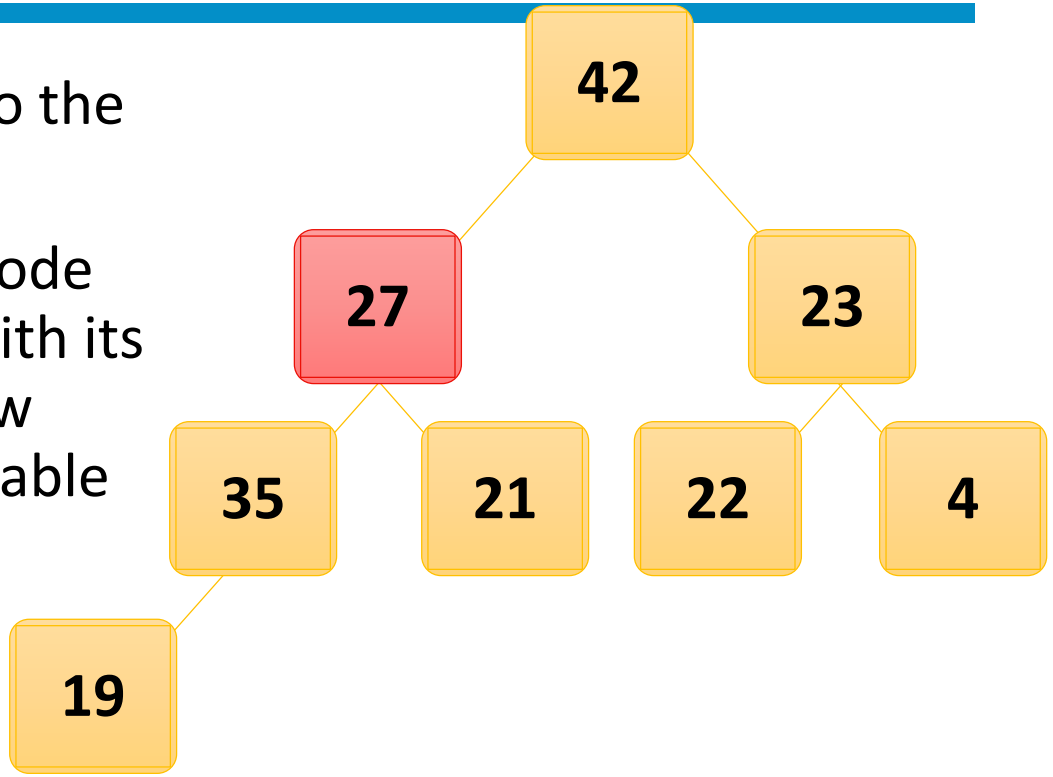
Removing the Top of a Heap

- Move the last node onto the root.
- Push the out-of-place node downward, swapping with its larger child until the new node reaches an acceptable location.



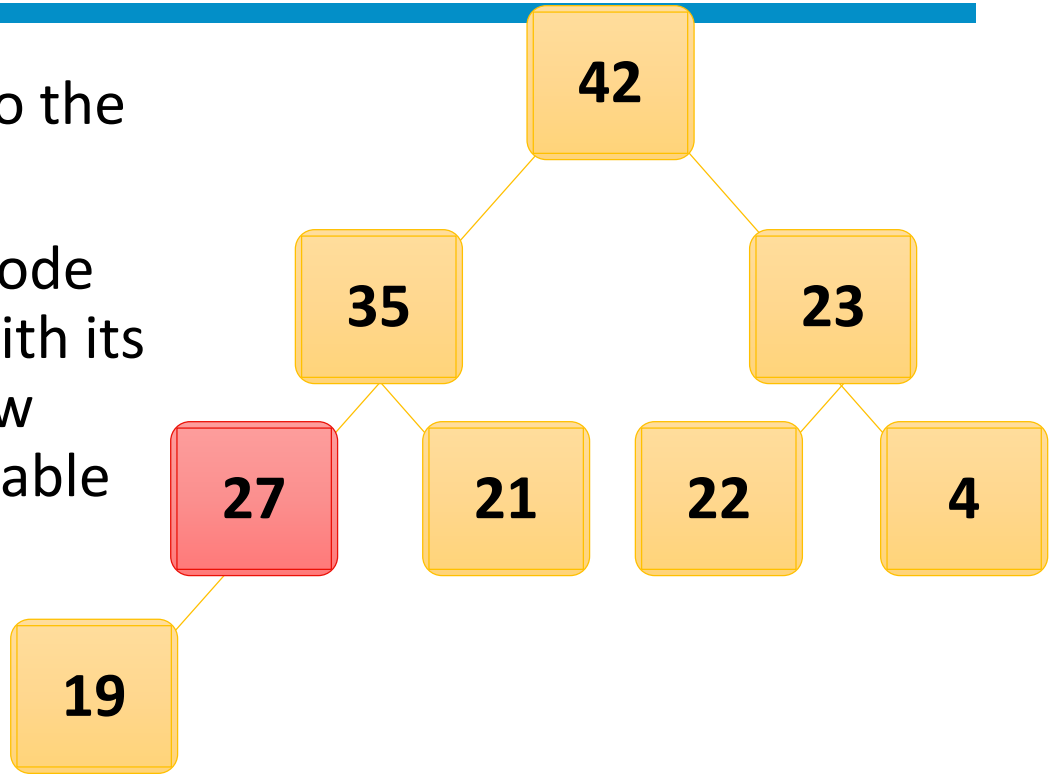
Removing the Top of a Heap

- Move the last node onto the root.
- Push the out-of-place node downward, swapping with its larger child until the new node reaches an acceptable location.



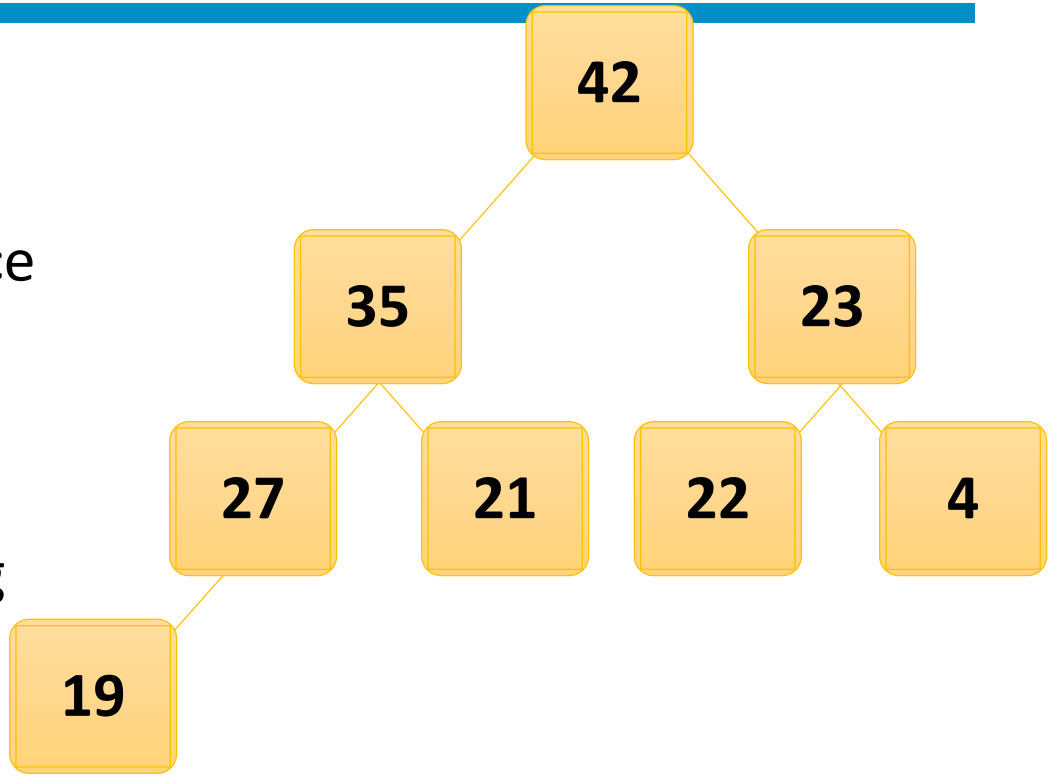
Removing the Top of a Heap

- Move the last node onto the root.
- Push the out-of-place node downward, swapping with its larger child until the new node reaches an acceptable location.



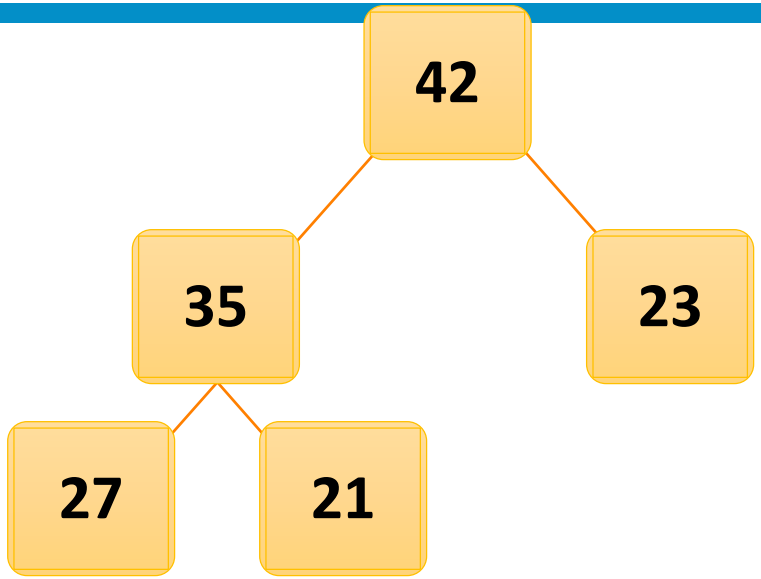
Removing the Top of a Heap

- The children all have keys <= the out-of-place node, or
- The node reaches the leaf.
- The process of pushing the new node downward is called **reheapification downward**.



Implementing a Heap

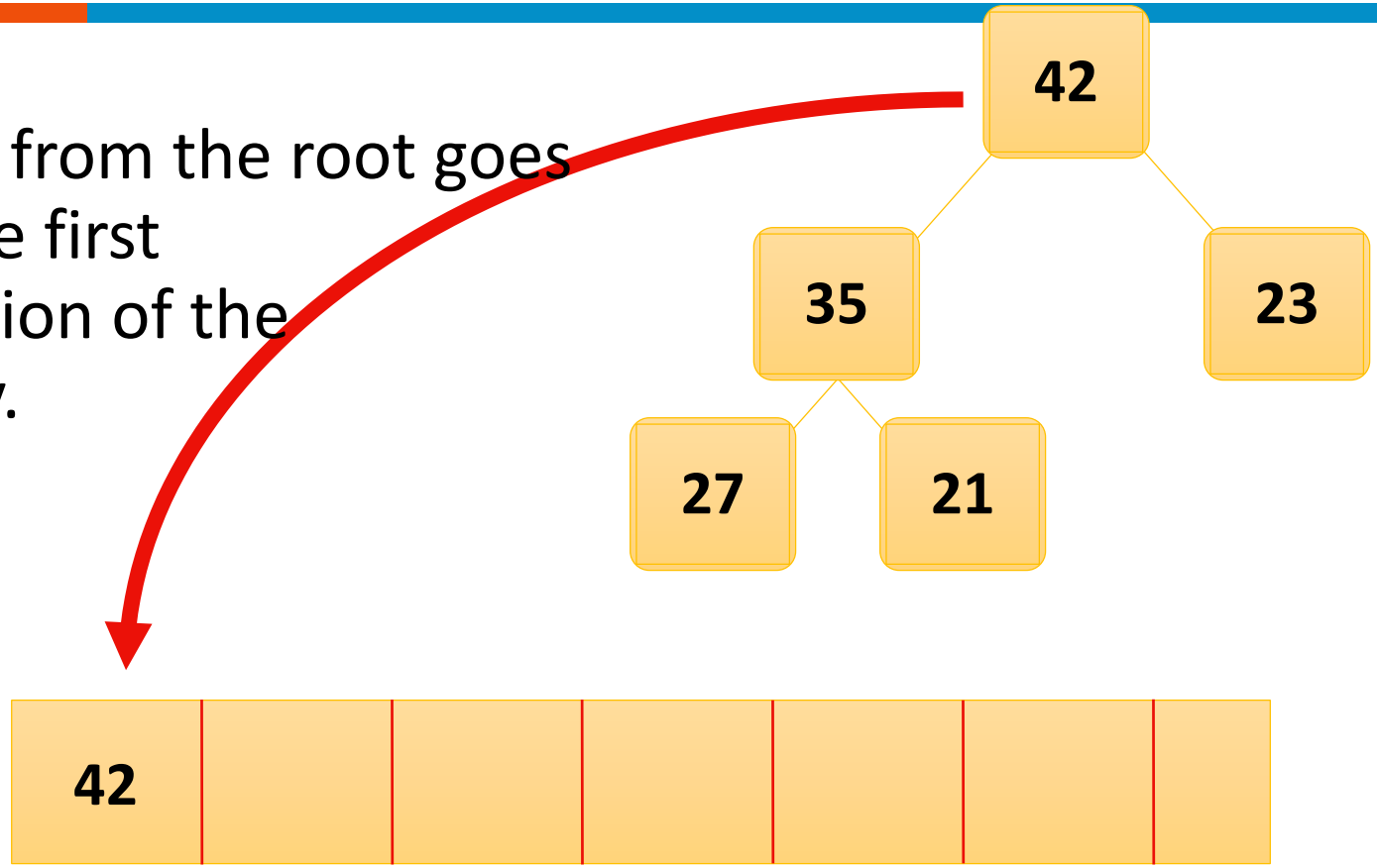
- We will store the data from the nodes in a partially-filled array.



An array of data

Implementing a Heap

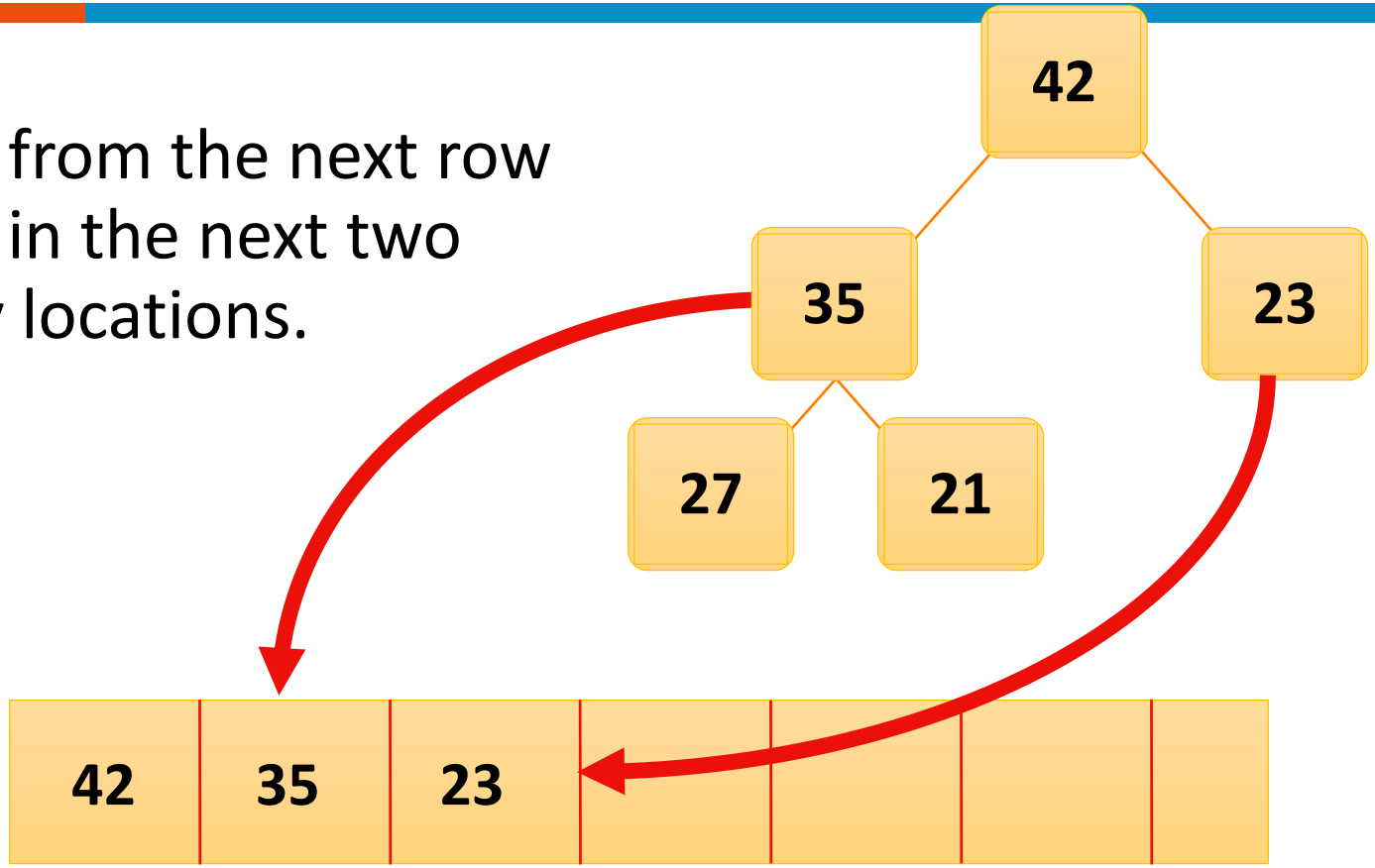
- Data from the root goes in the first location of the array.



An array of data

Implementing a Heap

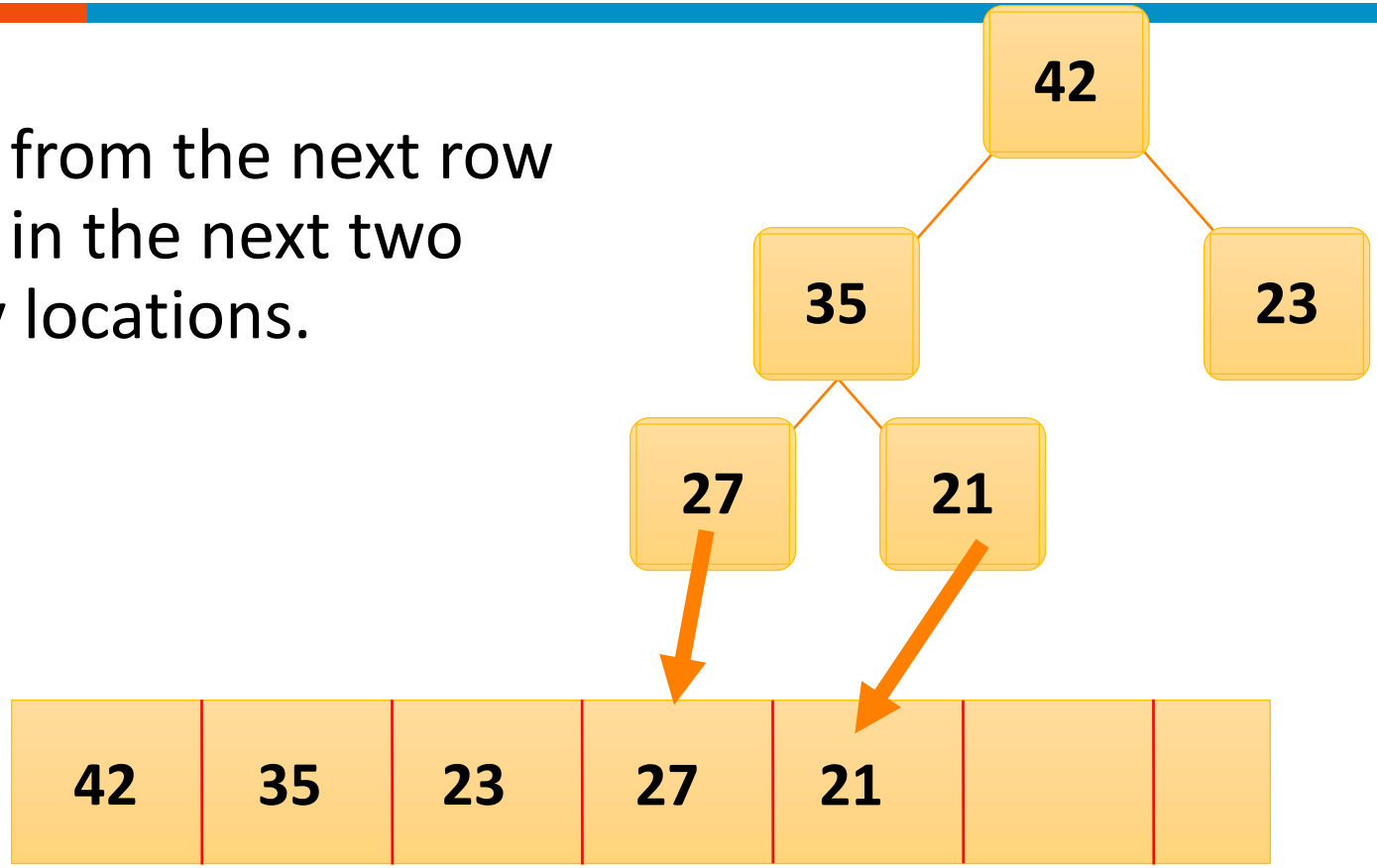
- Data from the next row goes in the next two array locations.



An array of data

Implementing a Heap

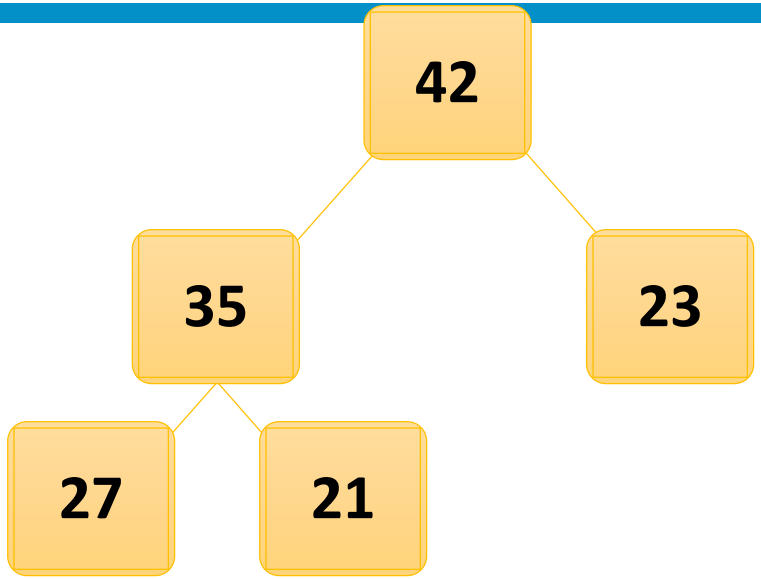
- Data from the next row goes in the next two array locations.

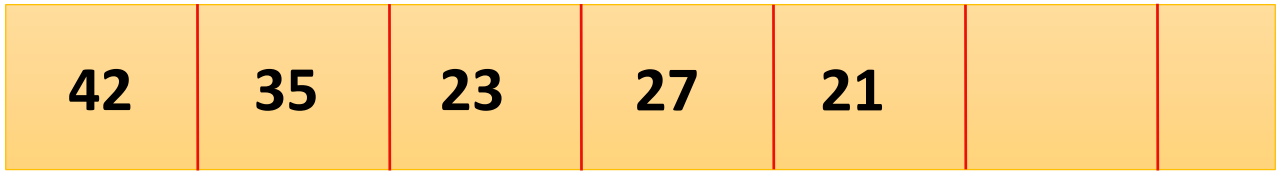


An array of data

Implementing a Heap

- Data from the next row goes in the next two array locations.



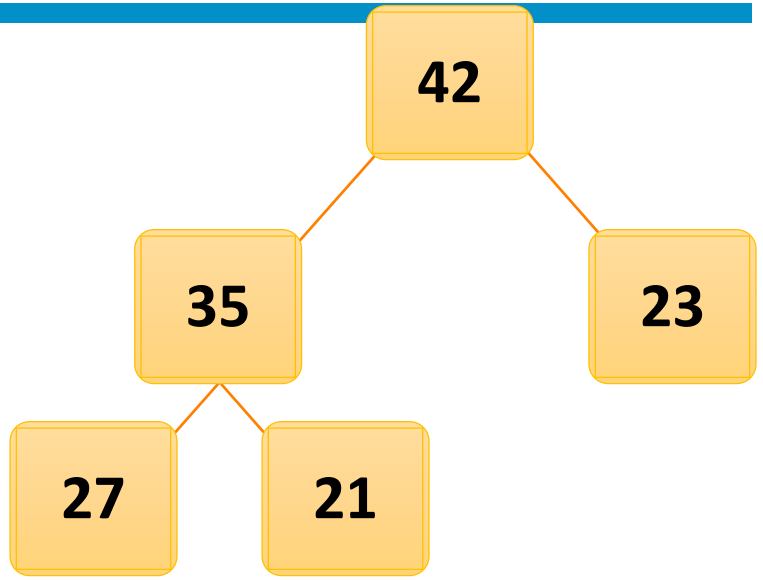


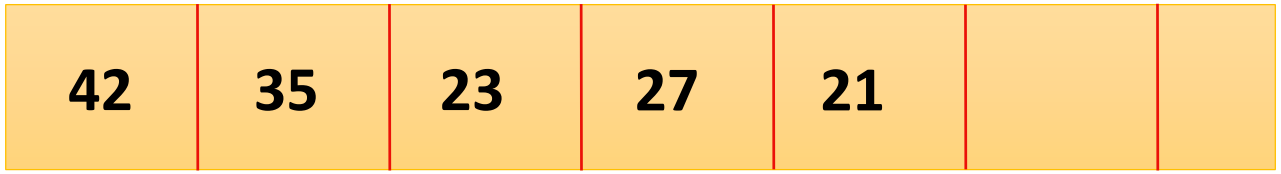
An array of data

We don't care what's in this part of the array.

Important Points about the Implementation

- The links between the tree's nodes are not actually stored as pointers, or in any other way.
- The only way we "know" that "the array is a tree" is from the way we manipulate the data.

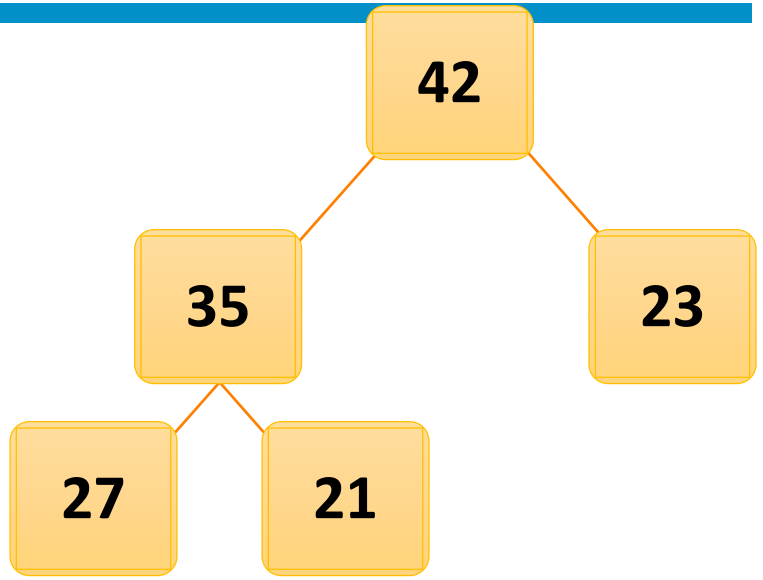


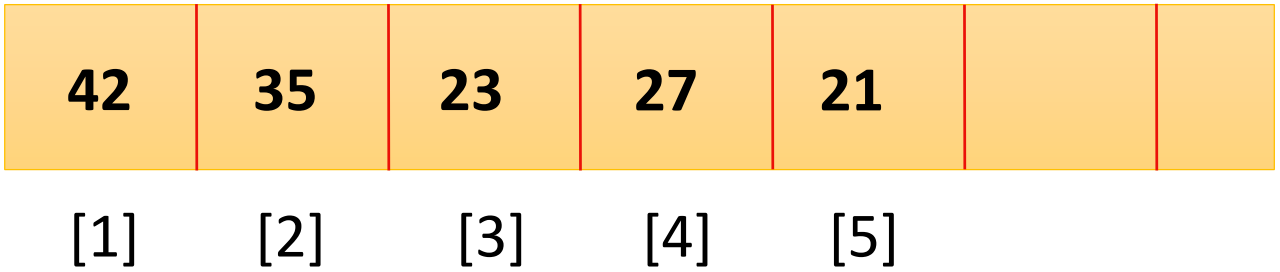


An array of data

Important Points about the Implementation

- If you know the index of a node, then it is easy to figure out the indexes of that node's parent and children. Formulas are given in the book.





Summary

- A heap is a complete binary tree, where the entry at each node is greater than or equal to the entries in its children.
- To add an entry to a heap, place the new entry at the next available spot, and perform a reheapification upward.
- To remove the biggest entry, move the last node onto the root, and perform a reheapification downward.