Heaps

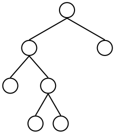
Specialized Binary Trees

The following special Binary Trees are NOT Binary Search Trees.

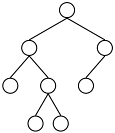
Two Tree

- Two Tree
  - A binary tree where every node has either 0 or 2 children

Two Tree

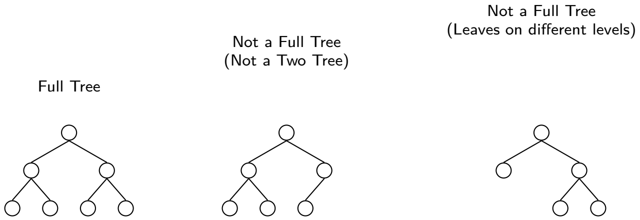


Not a Two Tree



Full Tree

- Full Tree
  - Two Tree with all leaves on the same level



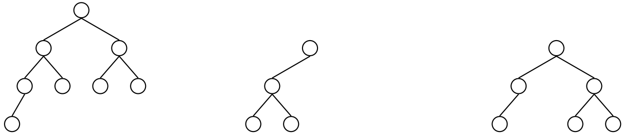
Complete Tree

- Complete Tree
  - A tree that is full up to the last level
  - The last level has all nodes as far left as possible

Complete Tree

Not a Complete Tree (Level 1 not full)

Not a Complete Tree (Last level not left)



- Every full tree is complete
- Not every complete tree is full
- A complete tree is NOT NECESSARILY a BST. There is no restrictions on the contents of the node

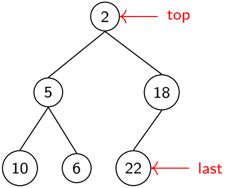
Complete Tree Properties

- Height is $\lfloor lg(n) \rfloor$
- Structurally representable as an array
  - More on this later

Heaps

- Heap
  - A complete binary tree with two properties:
    - All contents are directly comparable (numbers, for example)
    - For every node in the tree the contents are less than or equal to its descendants (Min Heap)
    - Alternately, a Max Heap will have every node be greater than or equal to its descendants
    - Discussion in these slides will focus on min heaps
- Recursive Definition (Min Heap)
  - Complete tree where:
    - The root of the tree has the smallest value
    - Both subtrees are min heaps

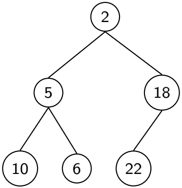
Heap Terms



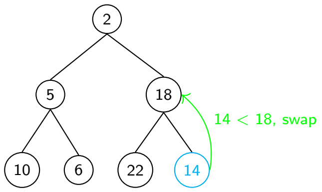
- The root of a heap is called the top of the heap
- The top is always the smallest element
- The last element in a heap is the rightmost position of the lowest level

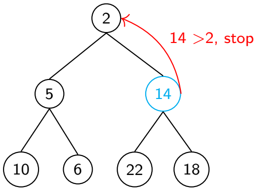
Heap Addition

- If the last level is not full, element is added to the next position available
- If the last level is full, element becomes first element in the new level
- Element is swapped up the heap if necessary
- Example: Add 14 to the heap



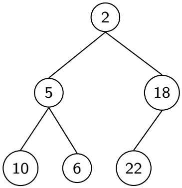
Example: Adding



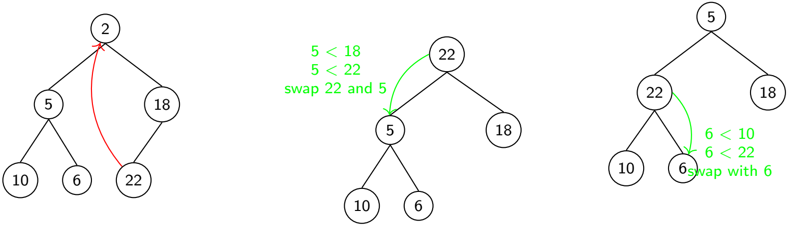


Heap Removal

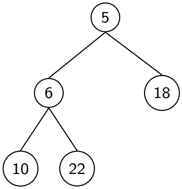
- Move the last element to the top
- Swap the new root element down the heap to maintain heapiness
  - For a min heap, swap with the smallest child element
- Example: Remove the top element



Example: Removing the Top

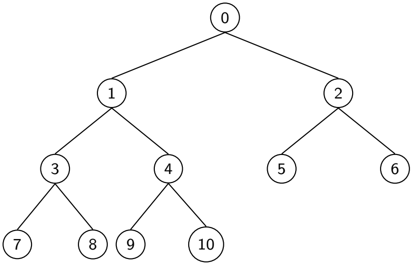


Removal Result



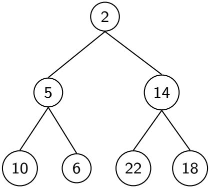
Complete Trees as Arrays

- Heaps can be implemented as an array or using binary tree nodes
  - Keeping track of last can be difficult with a binary tree
- In the below picture, values at nodes correspond to indexes in an array



Addressing

- Root has index 0
- For a node with index `i`, the left child has index `2*i+1`
- For a node with index `i`, the right child has index `2*i+2`
- For a node with index `i`, the parent has index `(i-1)/2` using integer division



2	5	14	10	6	22	18

Benefits

- Memory usage is low
- The last element has index array.size() - 1 (easy to keep track of)
- Insertion can be done using array.add(), which is easy and fast

Heap Performance

- Insertion into a heap is ok
  - Not as fast as insertion into a set or end of a list
  - Faster than inserting near the beginning of an array list
- Removal from a heap is ok
- Searching is slow (we only know something about the top of the heap)