**Heaps**

A type of tree, elements are ordered in increasing or decreasing order 🡪 Root is either MAX or MIN

* Max Heap: Parent always bigger than child (Root == "Peek")
* Min Heap: Parent always smaller than child

Heaps DON’T NEED to be binary trees

\*\*Binary Heap: Must be a "Complete" Tree - all levels except last one are completely full. Last level: values are added from left to right

**Search**: can only quit early if value is larger/smaller than current

🡪 average O(n/2) = O(n)

**Insert**: Still add to open spot, then Heapify: Compare with parent, then **swap**

**Extract**: root is returned, Heapify: root = right most leaf, then compare it with children, then **swap**

🡪 worst: move elements fully up or down 🡪 height of tree ~ O(log(n))

**Heap Implementation**: Sorted Array! Since we insert from left to right, and we know number of elements in a level (1, 2, 4, 8…) 🡪 Can keep track with array

A picture containing diagram

Description automatically generated

**Self-Balancing Tree**

**Red**-**Black** Tree

- Must have a color: either **black** or **red**

- Must have null **black** leaf nodes

- **Red** nodes must have 2 black children

- (Optional) Root must be **black**

- Every path from a node to its descendant NULL node must contain the same number of **black** nodes