Heaps

Definition 0.1

Heaps have the shape property of a binary tree (having 0-2 children), but heaps must also be *complete*.

This property is what makes heaps easy to implement with arrays

Min Heap

Definition 0.2

In **minheap**, the smallest data lives in the root and each child is greater than its parents value. There is no relationship between siblings

Heaps can be illustrated as an array given their characteristics, displayed in level order.

Definition 0.3

Given data at index n:
- Left child: $2 * n$
- Right child: $2 * n + 1$
- Parent: $\frac{n}{2}$

Example 0.1

Heaps Use Cases
- Not designed for arbitrary searching, mostly designed for accessing root
  - They are no better than just searching an arraylist (O(n))
- Heaps are often used to back priority queues

Heap Operations

Add Algorithm

- Add to the next spot in the array to maintain completeness
  - this would be index[size]

    Note 0.1

    We do not use **index zero** for heaps

- Up heap starting from the new data to fix order property
  - Compare the data with the parent, and swap the data with the parent as necessary until we read the top or no swap is needed. *This differs for min heap and max heap.*

Note 0.2

Time complexity of adding a new element is **O(logn)**. While adding is O(1), the upheap process is O(logn).

Remove Algorithm

- Move the last element of the heap to replace the root
- Down heap starting from the root to fix the order property. If two children, compare data with higher priority child.