Priority Queue ADT/Binary Heap data structure

Heap Property
\[ \text{Parent's value} \geq \text{children's value} \]

Structural Property - Perfectly balanced

Heap Sort
Perfectly Balanced

$n=1$

$n=2$

$n=3$

$n=4$

$n=5$

$n=6$

height $\leq \lceil \log_2 n \rceil$
Instead of using a tree rep. internally, instead use an array.

\[ \text{index} \]

\[ \text{n elements} \]

\[ \text{(if we know)} \]

\[ \text{tree} \]

\[ 4n \text{ refs} \]

\[ \text{array} \]

\[ n \text{ refs} \]

\[ \text{left}(i) = 2i + 1, \text{right}(i) = 2i + 2, \text{parent}(i) = \lfloor (i-1)/2 \rfloor \]
Insertion

Insert 9

array heap
that holds
\forall \ n \ # \ element

heap[n] = new item

while current item
is > parent
swap with
parent

Swap
Extract Max

Diagram of a tree structure with nodes labeled t, s, r, a, b, i, o, n, and arrows indicating connections. The diagram also includes a hand-drawn tree with letters s, t, c, r, a, b, a, i, o.
Remove an arbitrary element when given its index
Fix Upward (i)

was used in insertion

starting at index i

Fix Downward (i)

used in extract Max

) generally called heapify
Tracked Tagged Binary Heap

- put((tag, element))
  - returns a Tracker → putTracker((tag, element))

- Tracker = addTracker(0, e)
Application

Diagram with various nodes and arrows indicating connections and relationships.