**Implementing a Heap**

Because a heap is a complete binary tree, it can be implemented efficiently using an array rather than a linked data structure.

A picture containing schematic

Description automatically generated

For a node at position p,

* Left child position : 2p + 1
* Right child position : 2p + 2

For a node at position p,

* Parent position : (p-1) / 2 🡪 integer division, after the comma is ignored

Inserting into a Heap Implemented as an ArrayList

1. Insert the new element at the end of the ArrayList and set child to table.size() - 1
2. Set parent to (child - 1) / 2
3. while (parent >= 0 and table[parent] > table[child])

Swap table[parent] and table[child]

Set child equal to parent

Set parent equal to (child - 1) / 2

* Adding at the end of the ArrayList 🡪 amortized constant time
* Swapping parent and child 🡪 constant time
* while loop 🡪 inside is constant & loop is executed 0 times for the best case ( -- inserted value is larger than parent) and 1-logn times (there are logn levels and we have to do swapping at each level) for the worst case ( -- inserted element is the smallest in the queue)

Diagram

Description automatically generated with low confidenceA picture containing text, clock

Description automatically generatedDiagram

Description automatically generated with medium confidence

Diagram

Description automatically generated with medium confidenceA picture containing diagram

Description automatically generatedA picture containing diagram

Description automatically generated













Removal from a Heap Implemented as an ArrayList

Text

Description automatically generated

* First 2 lines and everything in the loop is constant time
* While loop:
  + worst case, from root to leaf 🡪
  + best case,

Shape, icon

Description automatically generated



Assume blue nodes are smaller than x and green-blue is smaller than black-blue.

Everything under the green-blue is larger than x. In this case, blue one and green-blue one go 1 up. Then you stop bc everything under the green-blue is larger than x. So best case is

If we couldn’t find this example, all we can say is only (1).



A picture containing diagram

Description automatically generated

Performance of the Heap

remove() traces a path from the root to a leaf

insert() traces a path from a leaf to the root

This requires at most h steps where h is the height of the tree

The largest full tree of height h has -1 nodes

The smallest complete tree of height h has nodes

Both insert() and remove() are O(logn)