# Heaps

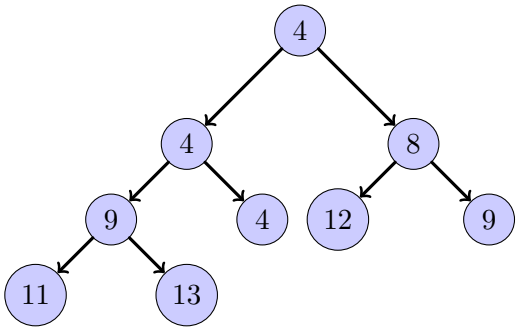
## 1 Properties

A heap is a generic container for objects with comparable keys. Inserting, deleting, and extracting the minimum or maximum (depending on the heap implementation) are all $O(\log n)$ operations, and heapifying an entire list is an $O(n)$ operation.

**Heapsort** Heapsort is a very simple $O(n \log n)$ sort that heapifies a list to be sorted, then extracts the minimum from the top of the heap until the heap is empty.

## 2 Implementation

Conceptually, a binary heap is a rooted binary tree that is as complete as possible. In a min-heap, at every node `x`, `key[x]` $\leq$ all child keys.




A binary heap can be intuitively represented by an array. The above heap would be represented as `[4 4 8 9 4 12 9 11 13]`. The relationship between array elements is as follows for a 0-based array.

- `parent(i)` $= (i-1)/2$
- `children(i)` $= (2i+1,\ 2i+2)$

### 2.1 Command Implementation

Every basic heap manipulation depends on two basic algorithms: bubble-up and bubble-down. Apart from that, it is possible to insert an element, extract the root, delete an element, and heapify a collection.

# Heaps

**Bubble Up** To bubble up, check if the heap property is violated by comparing an element with its parent. If it is, swap the two and repeat with the same element in the new position.

**Bubble Down** Compare the element with its children, and if the heap property is violated, swap the element with the smaller child, and repeat.

**Insert** Append the new element $k$ to the end of the array, and bubble up.

**Extract** Remove the root, move the last leaf node to the root, and bubble down.

**Delete** Remove the element, swap the last leaf node to its position, and bubble up or down as necessary.

**Heapify** Read data into an array, and starting from the lowest, rightmost parent node, bubble down.