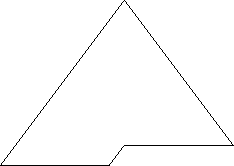
**The Heap structure**

The heap data structure is a binary tree with some properties.

**Heap-shapeproperty.** The tree is perfectly balanced; the leaves lie on at most two adjacent levels at the bottom and are pushed as far to the left as possible. All other levels are complete This property is illustrated graphically as



Thus, the heap has no holes and the leaves are at most two levels of the tree. The height of a tree representing a heap with *N* elements is *log2N*.

**Storage**. The elements of the heap can be stored implicitly (i.e. without additional overhead) in the *x[1..N]* array  with the children of the element *x[i]* being the elements at *x*[2*i*] and *x*[2*i*+1]. The parent of an element at position *k* can be found at *k/2* (integer division). The levels numbered from 1.

This structure the heap uses only the absolutely required to store the elements, and the shape property is guaranteed to hold.

**The MinHeap**

Min Heaps are often used for implementing priority queues. One of the earliest references can be found in [Williams] who uses a heap structure for sorting an array in *logN* worst case time (quicksort is quicker in practice).

**Min-ordering*(*Max-ordering*)* property**. the value stored at a node is smaller (greater) or equal than any of its descendants if they exist. Thus the root holds the smallest (largest) element.

The min-ordering defines a MinHeap, the max-ordering gives raise to a MaxHeap.

MinHeaps & MaxHeaps are useful for analyzing sorting algorithms and can be found in many proprietary queue implementations. They are not the most efficient algorithm as binomial trees [binomial] can do merging and XX in better asymptotic time.

**The MinMax Heap**

Described in [atkinson].

A MinMax Heap preserves the traditional shape property, but the Min-ordering property is replaced by the Min-Max Ordering property.

**Min-Max Ordering Property**. A tree is said to be min-max ordered if every element on even (odd) levels are smaller (greater) than or equal to the values stored in their descendants, if any.

**Corollary**. Any element on a max-level, i. e. an odd level, is larger than all of its descendants while any element on a min level is less than all of its descendants. As a result the minimum element can be found at the root and the maximum element is in one of the root’s two children.

**The Fine MinMax Heap**

Described in [nath].

**Dual heaps, etc**

**Appendix**

Properties of various heap structures:

|  |  |  |
| --- | --- | --- |
| **Operation** | **MinMaxHeap** | **MinHeap** |
| **Construction** | N? | N |
| **Insert** | LogN | LogN |
| **FindMax** | 1 | N |
| **FindMin** | 1 | 1 |
| **DeleteMin** | LogN | LogN |
| **DeleteMax** | LogN | LogN |
| **AdjustKey** | LogN ? | LogN ? |
| **Merge** | N ? | N ? |
| **Usage** | Double ended priority queue | Priority queue |

AdjustKey changes the value of an element at position *k* and rebuild the heap structure afterwards.

Merge combines two heaps.

**References**

[nath] Nath, Chowdhury, Kaykobad - MinMax Fine Heaps, XXX

[atkinson] M. D. Atkinson, J.-R. Sack, N. Santaro, and T. Strothotte Min-Max Heaps and Generalized Priority Queues, Communications of the ACM Oct. 1986 p. 996

[Williams] J. W. J. Williams, Algorithm 232, Heapsort, Communications of the ACM **7** (1964), 347-348

[binomial] binomial trees