heap insertion algorithm

Canonical name	HeapInsertionAlgorithm
Date of creation	2013-03-22 12:29:27
Last modified on	2013-03-22 12:29:27
Owner	mps (409)
Last modified by	mps (409)
Numerical id	9
Author	mps (409)
Entry type	Algorithm
Classification	msc 68P20
Classification	msc 68P10
Classification	msc 68P05
Related topic	Heap
Related topic	HeapRemovalAlgorithm
Related topic	Heapsort

The *heap insertion algorithm* inserts a new value into a heap, maintaining the heap property. Let H be a heap, storing n elements over which the `http://planetmath.org/Relation`relation $\preceq$ imposes a total ordering. Insertion of a value x consists of initially adding x to the bottom of the tree, and then *sifting* it upwards until the heap property is regained.

Sifting consists of comparing x to its parent y. If $x \preceq y$ holds, then the heap property is violated. If this is the case, x and y are swapped and the operation is repeated for the new parent of x.

Since H is a balanced binary tree, it has a maximum depth of $\lceil \log_2 n \rceil + 1$. Since the maximum number of times that the sift operation can occur is constrained by the depth of the tree, the worst case time complexity for heap insertion is $\mathcal{O}(\log n)$. This means that a heap can be built from scratch to hold a multiset of n values in $\mathcal{O}(n \log n)$ time.

What follows is the pseudocode for implementing a heap insertion. For the given pseudocode, we presume that the heap is actually represented implicitly in an array (see the binary tree entry for details).

HeapInsert$(H, n, \preceq, x)$
Input: A heap $(H, \preceq)$ (represented as an array) containing n values and a new value x to be inse
Output: H and n, with x inserted and the heap property preserved.
Procedure:
$n \leftarrow n + 1$
$H[n] \leftarrow x$
$child \leftarrow n$
$parent \leftarrow n$ div 2
while $parent \geq 1$ do
    if $H[child] \preceq H[parent]$
      then $child \leftarrow parent$
           $parent \leftarrow parent$ div 2
      else $parent \leftarrow 0$
    fi
od