Leftish Heap

Leftist Heap is similar to binary tree with two additional structural and ordering property:

- Like Binary Heap, it maintains heap-order property. Root is always smaller than or equal to its children.
- Does not have a balancing condition like balanced binary tree. In fact, Leftist Heap propagates towards unbalanced state. This is because for each node, the null path length (npl) of the left sub-tree is at least as much as that of the right sub-tree. Hence the name $leftish$ as the heap is skewed towards left.

Null path length

Null path length of any node X, $npl(X)$ is defined as the shortest path to a node with one or zero children. Thus, npl of a node with one or zero child is 0 and $npl(None) = -1$

Efficacy of the Leftist Heap stems from the fact that its right path is guaranteed to be short. It can be shown that the right path of a Leftist Heap is $O(logN)$. Thus all work $(insert, merge)$ is done on its right path.

Height of the right path is $O(logN)$

Consider incrementing the number of nodes r (starting from 0) for right path of a binary tree and figuring out how many additional nodes must be added to make it Leftish. In the illustration below, additional nodes are shown inside brackets.

```
r   Tree        Leftish Tree     Additional nodes       Minimum Total nodes
-------------------------------------------------------------------------
0   1               1               0                       1

1   1               1               1                       3
     \             / \
      2          (3)  2

2   1                 1            4                       7
     \               /   \
      2             (4)     2
       \           /  \    /  \
        3         (5) (6) (7)  3
```

Following the pattern, for r nodes in the right path ($r + 1$ including root), at least $2^{r+1} - 1$ nodes are needed for the tree to be Leftish. Thus,

$$N \geq 2^{r+1} - 1$$

$$r + 1 \leq log(N + 1)$$

$$r = O(logN)$$

This guarantees $O(logN)$ complexity of $merge$ and $insert$!