**Leftist Heaps**

***null path length*** of a binary tree is the shortest path from the node to a node without two children

private Node<E> merge( Node<E>t1, Node<E> t2) {

Node<E> small;

if (t1==null) return t2;

if (t2==null) return t1;

if (t1.element.compareTo( t2.element ) < 0) {

t1.right = merge(t1.right, t2);

small=t1;}

else {

t2.right = merge(t2.right, t1);

small=t2;}

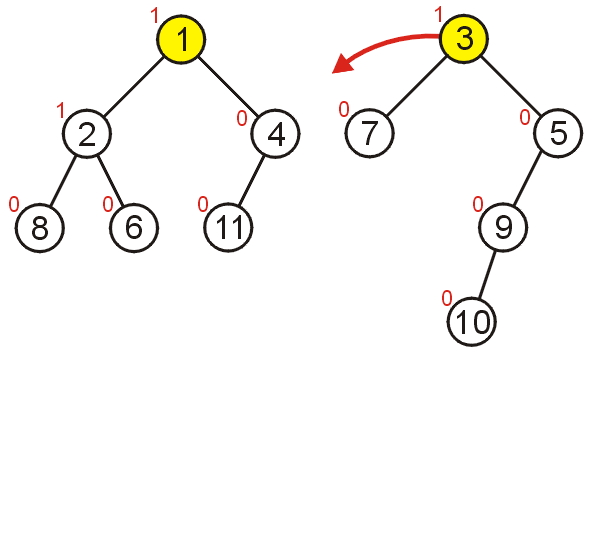
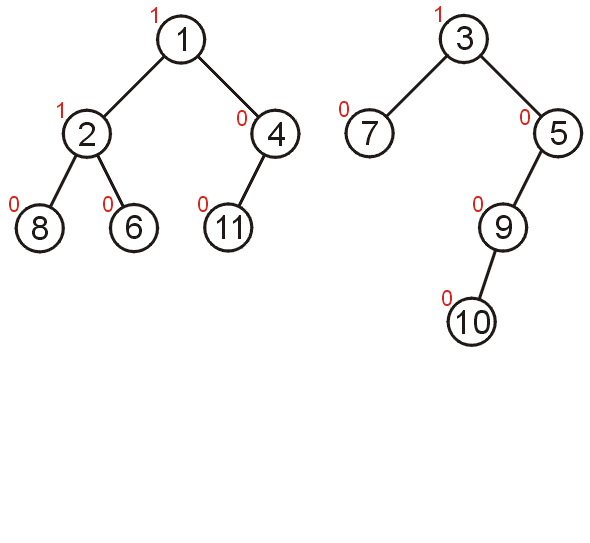
if (notLeftist(small)) swapkids(small);

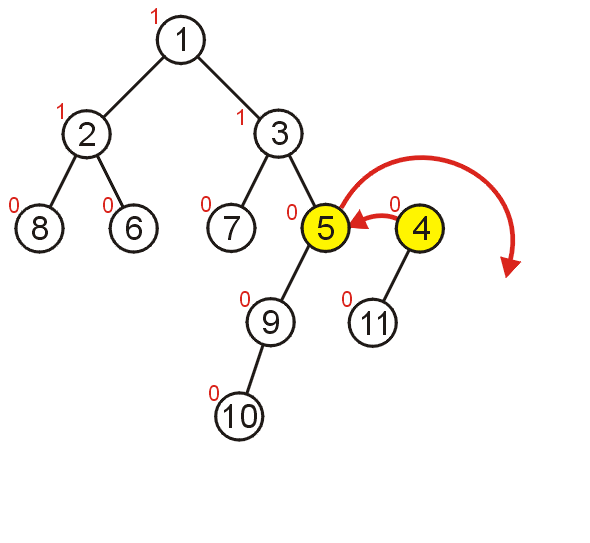
setNullPathLength(small);

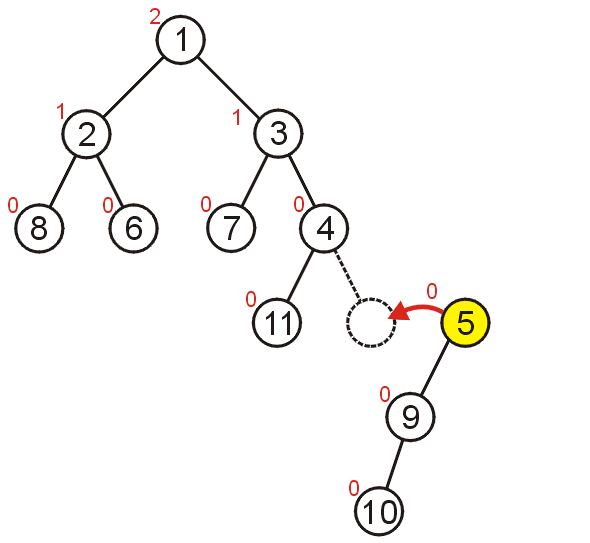
return small;

}

Consider merging these two leftist min heaps (null path lengths are indicated in red)





* On the way back out of the recursion, swap sub-heaps where necessary. Find the unhappy nodes – after updating the null path lengths.

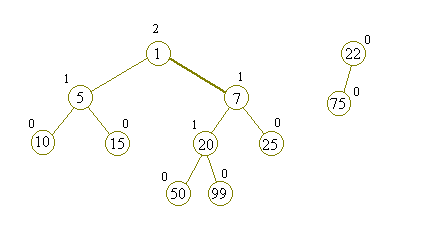
|  |  |
| --- | --- |
| a5 | a6 |

* Insert**:** Inserting merges the existing heap with a heap of size one
* DeleteMin: removes the root, and merges the children.

|  |  |
| --- | --- |
| a7  **Delete Min** | a0 |
| a1 |  |

|  |  |
| --- | --- |
| a3 | a4 |
| a7 | a8 |

Show the merging of the following heaps:



Show delete min of the following tree:

