Binomial heaps

Amogh Karhadkar
1BM18CS014

```
Insert ( heap, value)

    create new node with value as value
    create temperory heap
    inserting at the end of the heap
    looping over the heaps → original and temperory until one becomes NULL
        if degree of original tree in heap is less than
            degree of temperory tree in heap
                create a new heap and add the original tree.
        else
                add the temperory tree to heap

    if original heap has left over trees
        loop over and add them to
            new heap

    if temperory heap has left over trees
            loop over and add all of
            them to new heap

    if heap size is less than 1 return the heap

    loop over new heap
        if its end of heap only
        one element remains
        else if degree first tree less than degree
            of second tree then merge the trees
```

[illegible]

```
else if the degrees are same then
        Binomial tree are same in heap

else if degree of two binomial tree are
        same in heap then merge
        the trees

return the heap


getMin (heap)
    start from the first tree in
    the heap and keep checking
    the root of the trees.
    if less than lowest store it
    return the least value.


extractMin (heap)
    get the minimum value from getMin
    start from the first tree in the heap
    if the tree.root is not minimum
    then create a new heap and add
      the tree to the heap
    remove the minimum element from the heap
    and concat the tree to heap
    . Merge the newly created heap without minimum
      element and the heap that was created earlier.
    return the merged heap
```

[illegible]