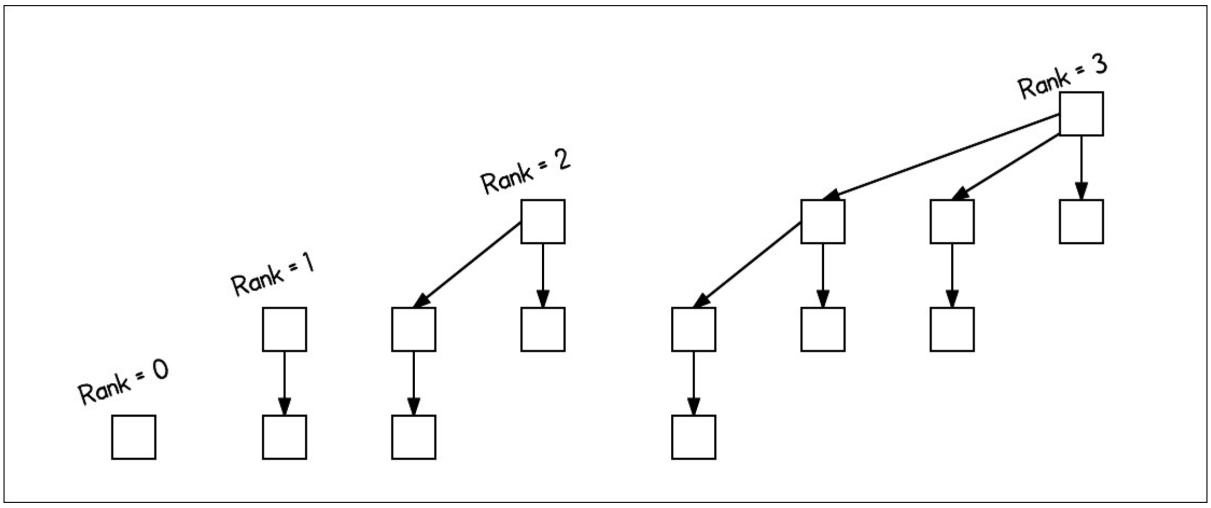
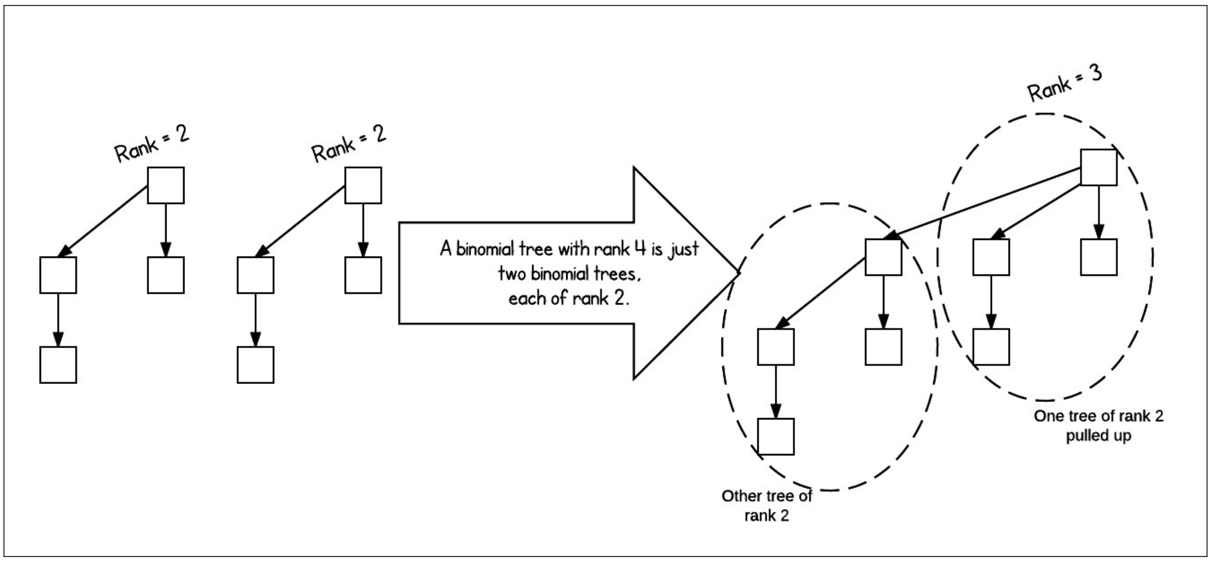
What is a Binomial Tree?

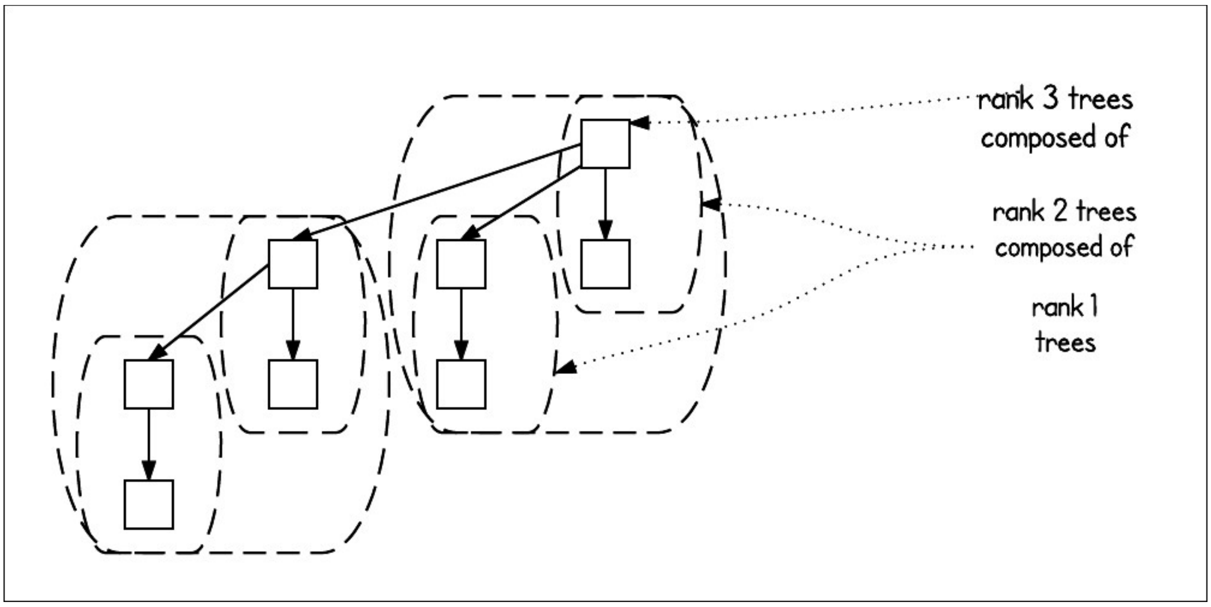
The Binomial tree Bk is the ordered tree made of linking two binomial trees, Bk-1 in which one becomes the leftmost child or the other. The number of nodes the zero-order binomial tree has is 1.

Some properties of the binomial tree are:

- It has 2^k number of nodes where k is the order.
- The tree has a depth equal to k.
- The children of the root, which has order k, are also binomial trees with orders k-1, k-2, and 0 from left to right.




Rank = 2
Rank = 2
A binomial tree with rank 4 is just two binomial trees, each of rank 2.
Rank = 3
One tree of rank 2 pulled up
Other tree of rank 2



Introduction to Binomial Heap

A binomial heap is a collection of binomial trees, each of which satisfies the heap property, i.e., the min-heap property. Each binomial tree is in heap order. So we can say the key of the node is greater than or equal to the key of its parent. There can be at most one binomial tree of any degree.

Properties of Binomial Heap?

The binomial heap has n nodes that should follow these properties:

- Each binomial tree in the heap should follow the min-heap property, i.e, the value of the node is greater than or equal to the value of its parent.
- At least one binomial tree should be in a heap where the root has a degree of k. where k can be any non-negative integer.
- First, ensure the min-heap property throughout the heap.
- there can be at most one Binomial Tree of any degree.

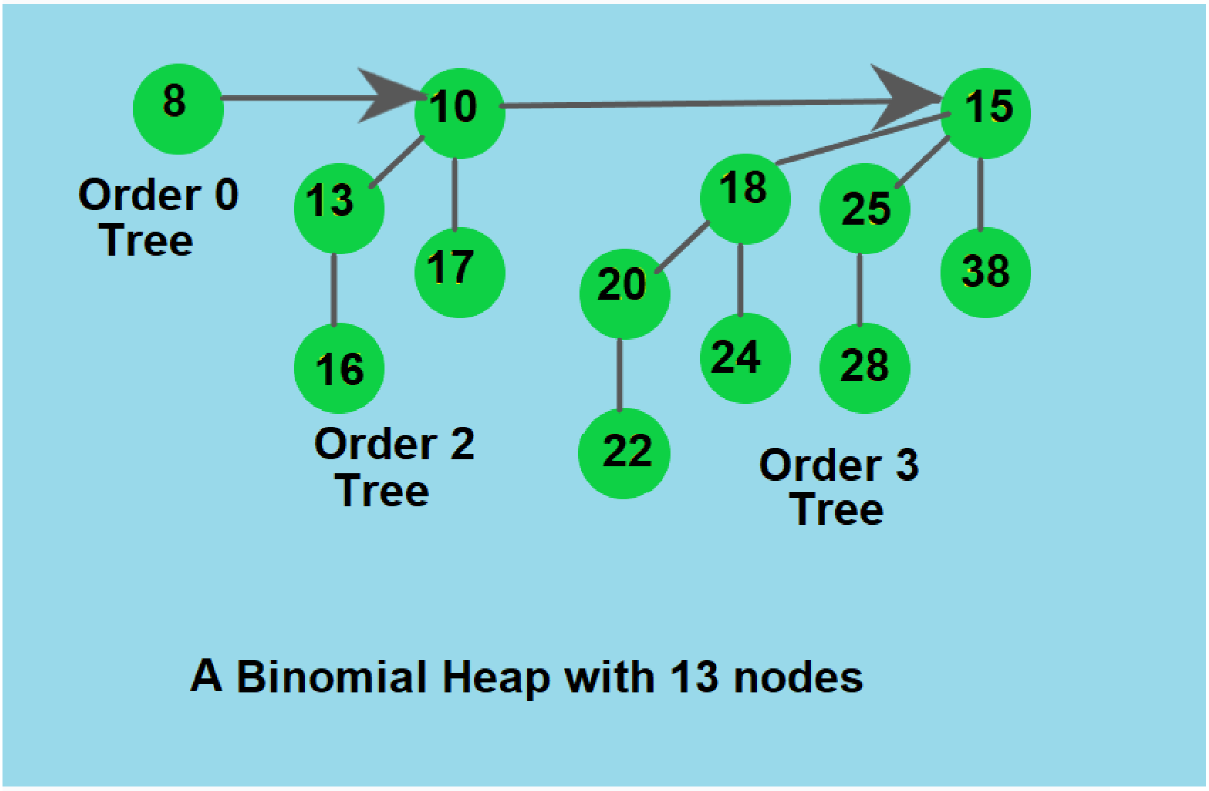
```
12------------10--------------------20
             /  \                 /  | \
           15    50             70  50  40
           |                  / |    |
           30               80  85  65
                            |
                           100
A Binomial Heap with 13 nodes. It is a collection of 3
Binomial Trees of orders 0, 2, and 3 from left to right
```

Binomial Heap and the Binary Representation of a Number

The binomial heap can be used to represent the binary number also, i.e., if the binomial heap has n binomial trees, where n is the number of set bits in the binary representation of the number.

If we want to create a binomial heap of n nodes, then it can be defined by the binary number 'n'. Let us explain with a plethora of examples. If we want to create a binomial heap of 15 nodes, the binary representation of 15 is 1111, so now numbering from the right-hand side, the set bits are at positions 0,1,2,3. Therefore, the binomial heap will be formed with 15 nodes and the binomial tree B0, B1, B2, and B3.

Let's try to understand with one more example. Suppose we want to create a binomial heap of 11 nodes. The binary representation of 11 is 1011, numbering from the right-hand side. The set bits are at 0, 1, and 3, so the binomial tree formed will be B0, B1, and B3. The degree of trees will be 0, 1, or 3 respectively.



Operations

The main operation in Binomial Heap is union(), all other operations mainly use this operation. The union() operation is to combine two Binomial Heaps into one.

The Basic operations of the Binomial Heap are as follows:

- Union(H1, H2) or Merge(H1, H2)
- Insert(k)
- GetMin()
- ExtractMin()
- DecreaseKey(x, k)
- Delete(x)

Union(H1, H2) or Merge(H1, H2)

The simplest and most important operation is the merging(union) of two binomial trees of the same order within a binomial heap. Due to the structure of binomial trees, they can be merged trivially. As their root node is the smallest element within the tree, by comparing the two keys, the smaller of them is the minimum key, and becomes the new root node. Then the other tree becomes a subtree of the combined tree. This operation is basic to the complete merging of two binomial heaps.

1. The first step is to simply merge the two Heaps in non-decreasing order of degrees. In the following diagram, figure(b) shows the result after merging.
2. After the simple merge, we need to make sure that there is at most one Binomial Tree of any order. To do this, we need to combine Binomial Trees of the same order. We traverse the list of merged roots, we keep track of three-pointers, prev, x and next-x.

3. There can be following 4 cases when we traverse the list of roots.
    - Case 1: Orders of x and next-x are not same, we simply move ahead.
      In following 3 cases orders of x and next-x are same.
    - Case 2: If the order of next-next-x is also same, move ahead.
    - Case 3: If the key of x is smaller than or equal to the key of next-x, then make next-x as a child of x by linking it with x.
    - Case 4: If the key of x is greater, then make x as the child of next.

Inserting a node

It is possible to insert an element into the heap by simply creating a new heap that contains the element to be added and merging it with the existing heap.

Extracting Minimum Key

eliminate an element with the smallest key value. As is common knowledge, the root element of a min-heap contains the smallest key value. Therefore, we must compare the root node's key value across all binomial trees

Decreasing a Key

Let's proceed to the subsequent operation on the binomial heap. Once the key's value is reduced, it may become smaller than the key of its parent, which constitutes a violation of the min-heap property. After lowering the key, if such a situation arises, swap the element with its parent, grandparent, and so forth until the min-heap property is met.

Delete a Node

The minimum node in the heap must be deleted to remove a node from the heap. To do this, we must first reduce the key of the node to negative infinity (or -). With the aid of an example, we'll now see how to delete a node.

Represent Binomial Heap

1. A binomial heap is a collection of binomial trees.
2. The binomial tree should be arranged or represented in such a way that it allows sequential access to all of its siblings from the leftmost sibling.
3. The key concept is to represent binomial trees with each node storing two pointers, one to the leftmost child and the other to the right sibling.