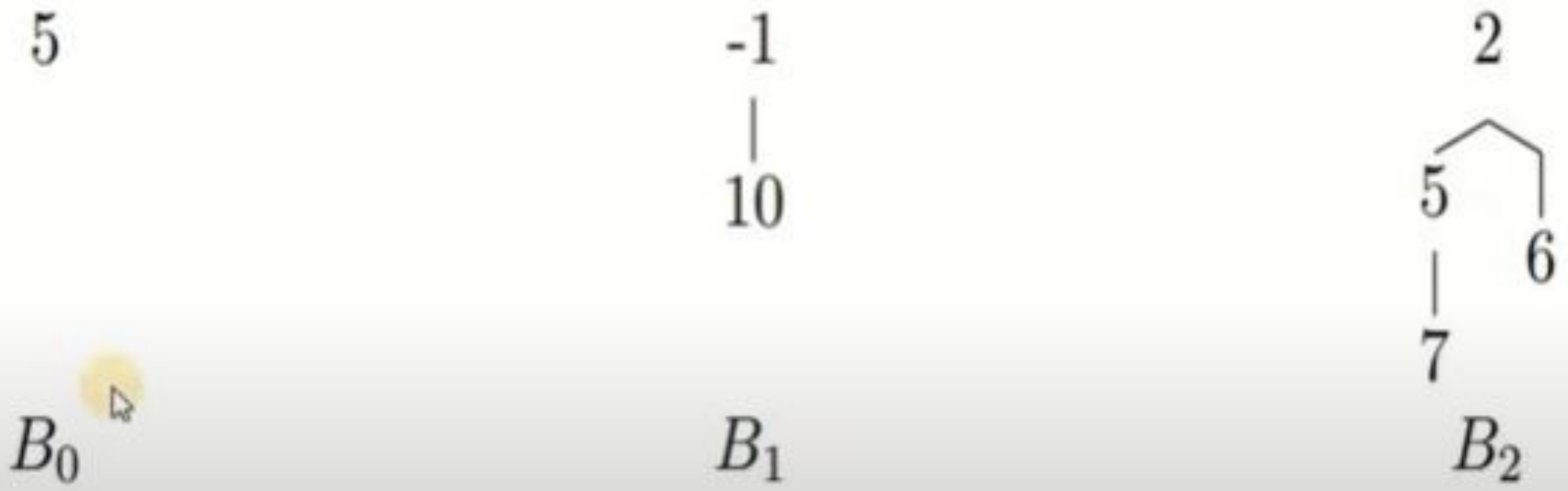
Binomial Heap

Heap (Binary Heap)

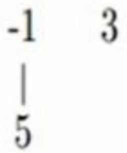
- Binary tree in which all nodes follow heap property
  - MinHeap: key(parent) <= key(child)
  - MaxHeap: key(parent) >= key(child)
  - All levels are completely filled except the last level, which is left filled
- Insertion - Add child at lowest level and shift up
- Deletion of root - Remove the rightmost leaf at the deepest level and use it for the new root and shift up

Binomial Tree

- A Binomial Tree B_k of order k is defined as follows
  - B_0 is a tree with one node
  - B_k is a pair of B_{k-1} trees, where root of one B_{k-1} becomes the left most child of the other (for all k ≥ 1)
  - two Bk−1's are combined to get one Bk, the Bk−1 having minimum value at the root will be the root of Bk, the other Bk−1 will become the child node.


5
-1
10
2
5
6
7
B_0
B_1
B_2

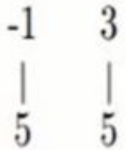
Example - [5 -1 3 5 7 8 9]



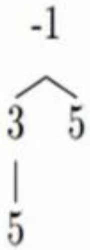
Insert 3 into B_1, we get one B_1 and a B_0.

Insert 5,

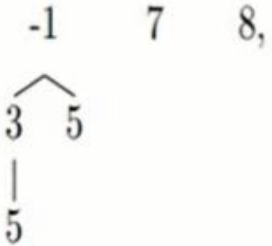
on merging two B_0's →



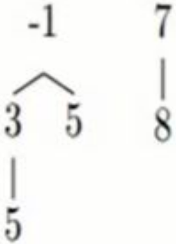
on merging two B_1's →



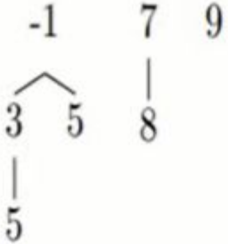
Insert 7, and 8.



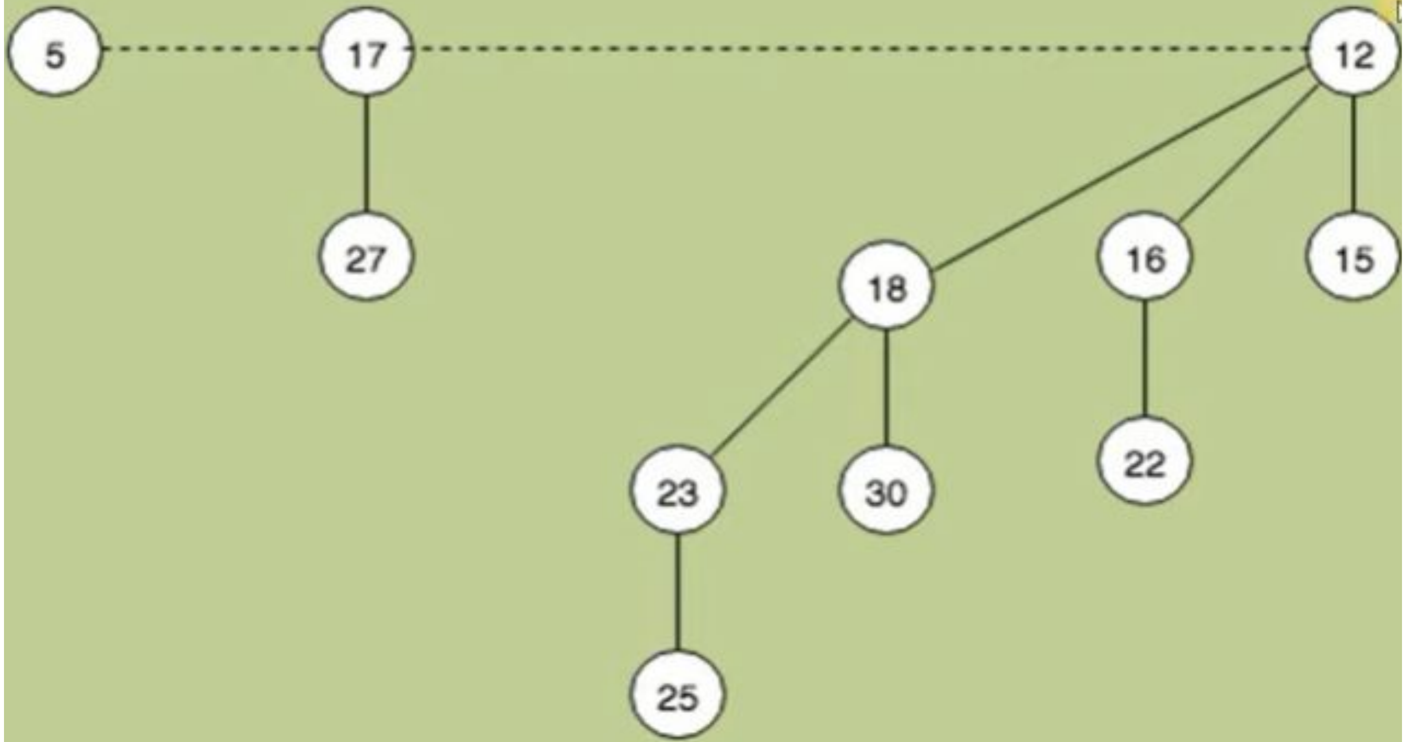
on merging two B_0's →



Insert 9,



Binomial Heap - Example

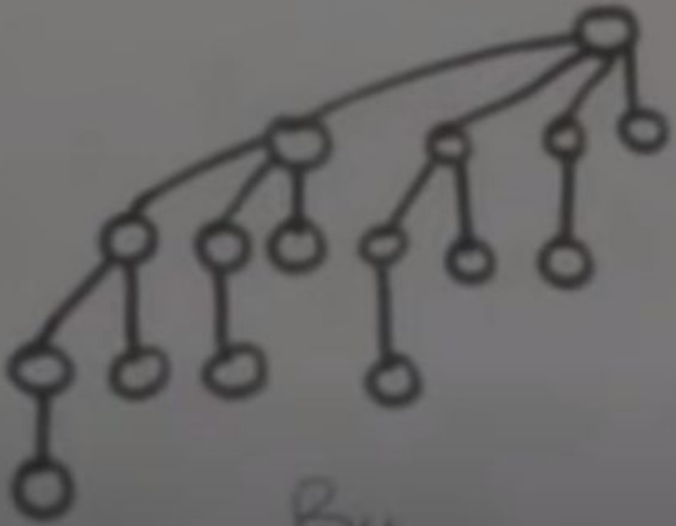


Structural Properties

For the binomial tree Bk.

- There are 2k nodes.
- The height of the binomial tree is k.
- There are exactly kCi nodes at depth i = 0, 1, . . . , k.
- The root has degree k, which is greater than that of any other node, moreover if the children of the root are numbered from left to right by k − 1, k − 2, . . . , 0, child i is the root of the Subtree Bi .

Note: Due to Property 3, it gets the name binomial tree (heap).



B4

depth

0 → ${}^4C_0 = \frac{4!}{4! \times 0!} = 1$

1 → ${}^4C_1 = \frac{4!}{(4-1)! \times 1!} = \frac{4!}{3! \times 1!} = 4$

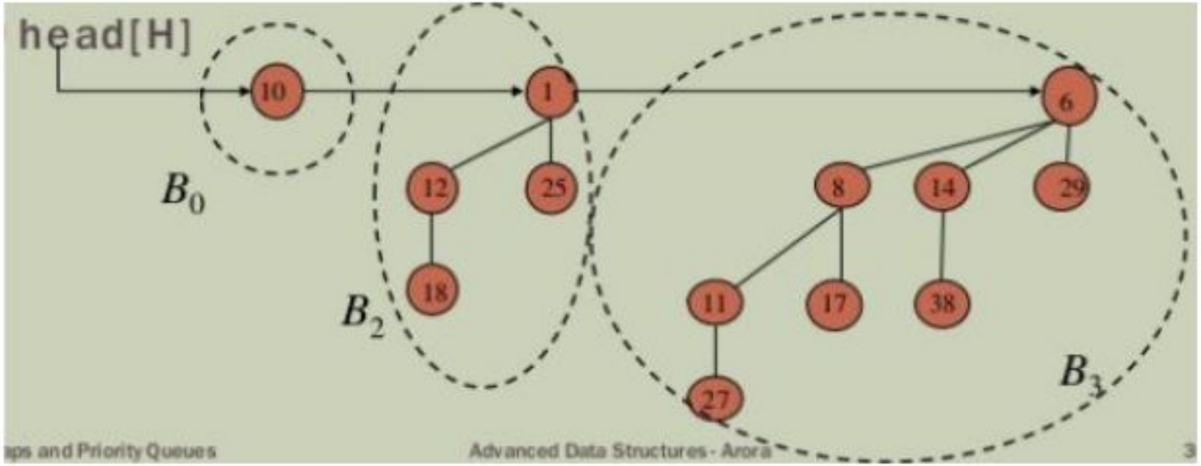
2 → ${}^4C_2 = \frac{4!}{(4-2)! \times 2!} = \frac{4!}{2! \times 2!} = 6$

3 → ${}^4C_3 = \frac{4!}{(4-3)! \times 3!} = \frac{4!}{1! \times 3!} = 4$

4 → ${}^4C_4 = \frac{4!}{(4-4)! \times 4!} = \frac{4!}{0! \times 4!} = 1$

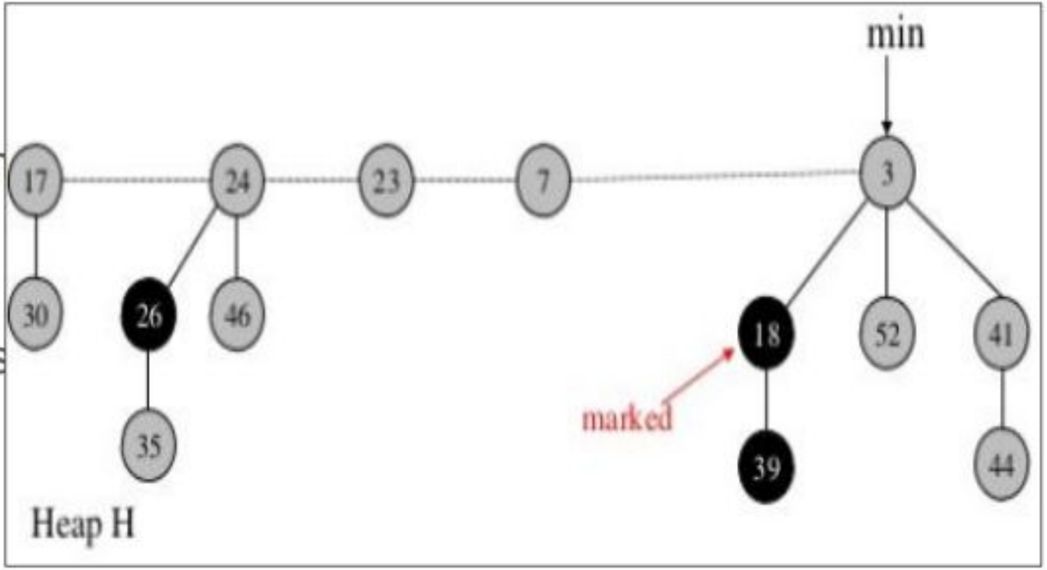
Binomial Heap

- Pointer points to the first node to enter into the heap
- The roots of the trees are connected so that sizes of the connected trees are in order



What is Fibonacci Heap?

- Collection of trees satisfying minimum-heap property i.e parent(value)<child(value)
- Maintains pointer to minimum element
- Contains set of marked nodes (to indicate if node has lost a child)
- Roots of all trees are linked using circular doubly linked list

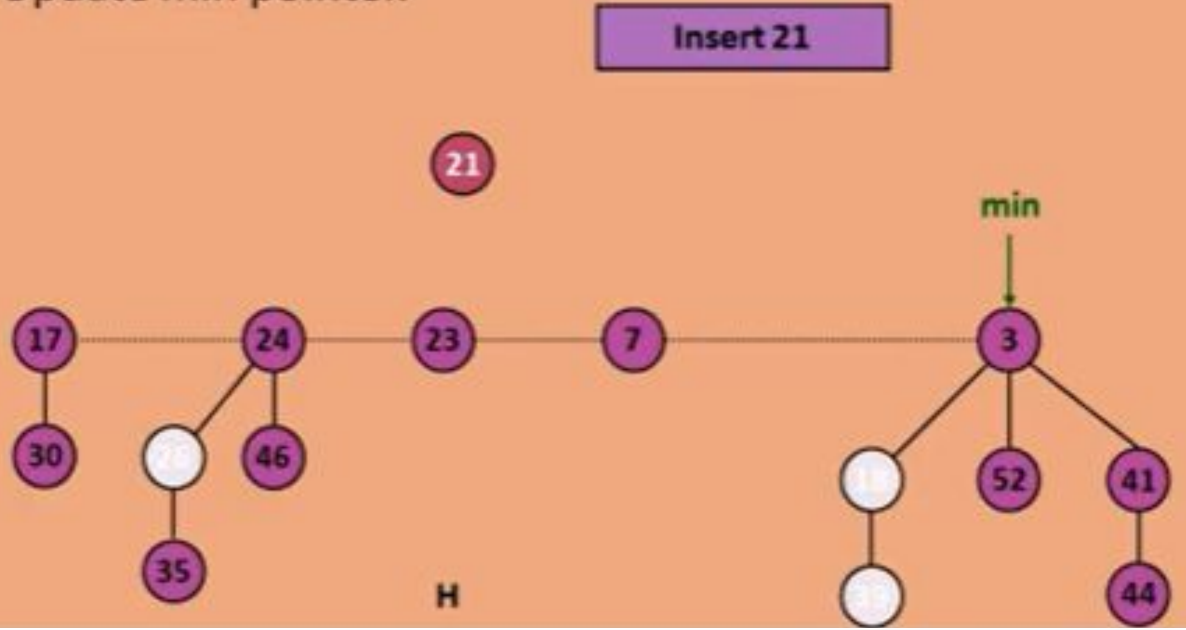


Fibonacci Heap Operations

- Creation
- Insertion
- Finding Minimum Key
- Union
- Extract Minimum Key
- Decrease Key
- Deletion

Fibonacci Heaps: Insert

- Insert.
  - Create a new singleton tree.
  - Add to left of min pointer.
  - Update min pointer.



Fibonacci Heaps: Insert

- Insert.
  - Create a new singleton tree.
  - Add to left of min pointer.
  - Update min pointer.

Insert 21