Binomial Heap

(i) Binomial Tree

↳ Binomial Tree is an ordered tree. It is not a Binary Tree. and It is represented with B_K.

B_K = Binomial Tree with order K.

Ex:-

B_0 → Binomial Tree with order 0.

i.e ○ = B_0

(This node having no children)

B_1 → Binomail Tree with order 1

i.e

B_1 [node has one child]

So, B_1 can be formed with two B_0 nodes i.e

B_1 =

Rules :-

* B_k has k children at root
* B_k can be formed using two B_{k-1} Trees
* The root of one B_{k-1} will be the Left most child of the root of other B_{k-1}

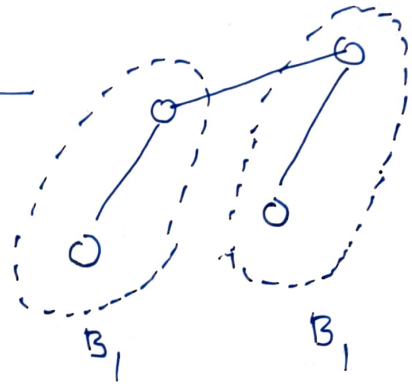
Ex :-

B_0 — O

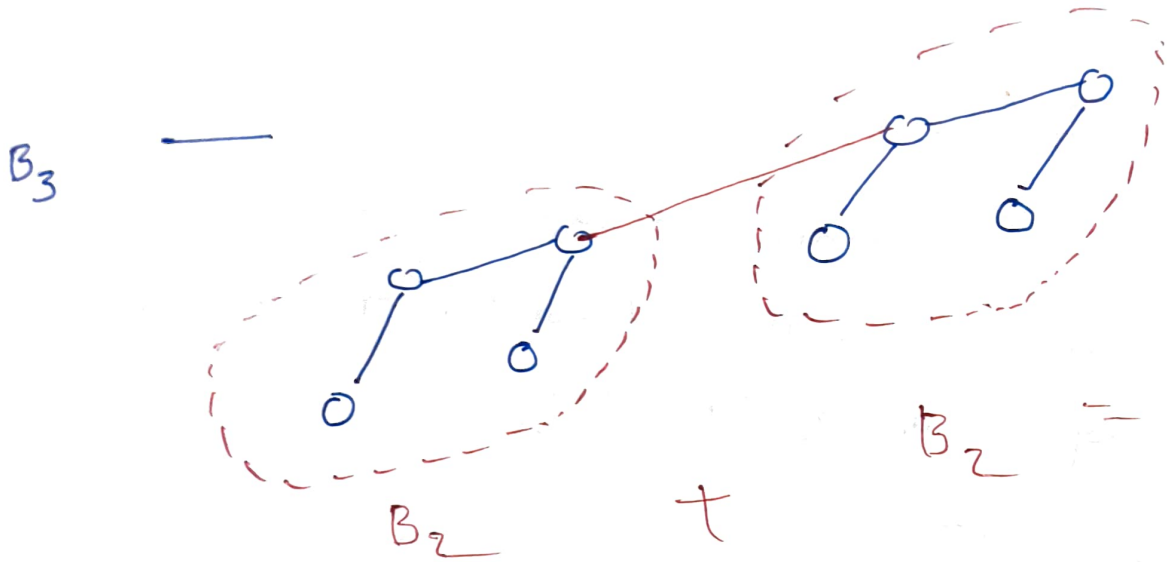
B_1 (k=1) — O B_0

O B_0

B_2 (k=2) —

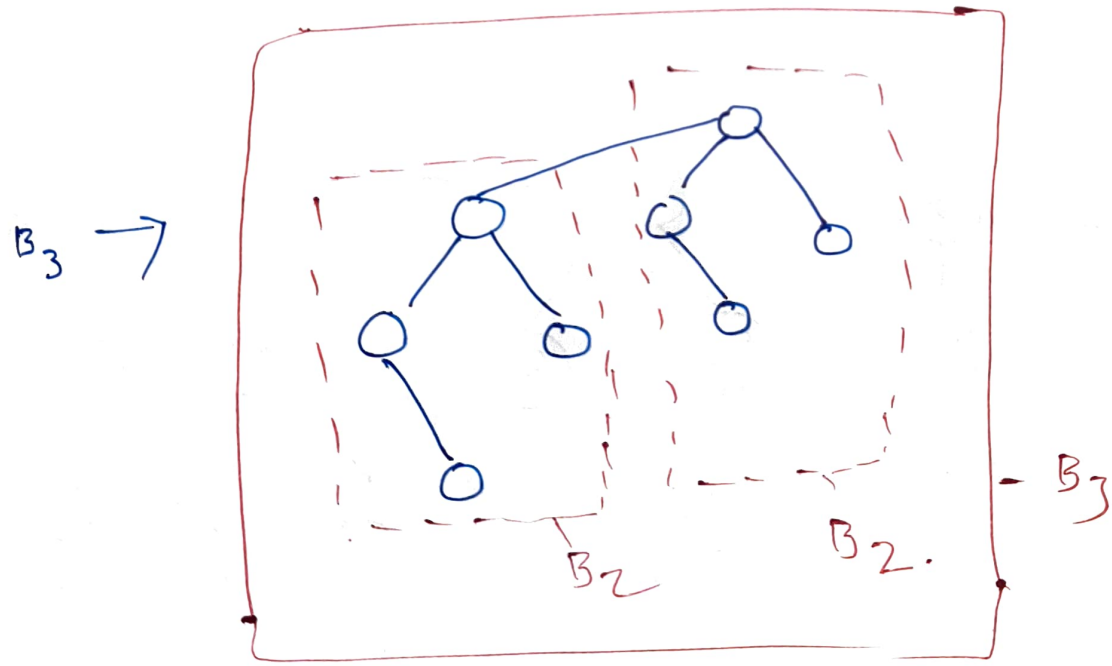


B_2 will be formed with Two B_1's



B_3 will be formed with Two B_2's

∴ Binomial Tree is



NOTE

B_4 have Two B_3's

B_5 have Two B_4's

!

B_k have Two B_{k-1}'s Trees

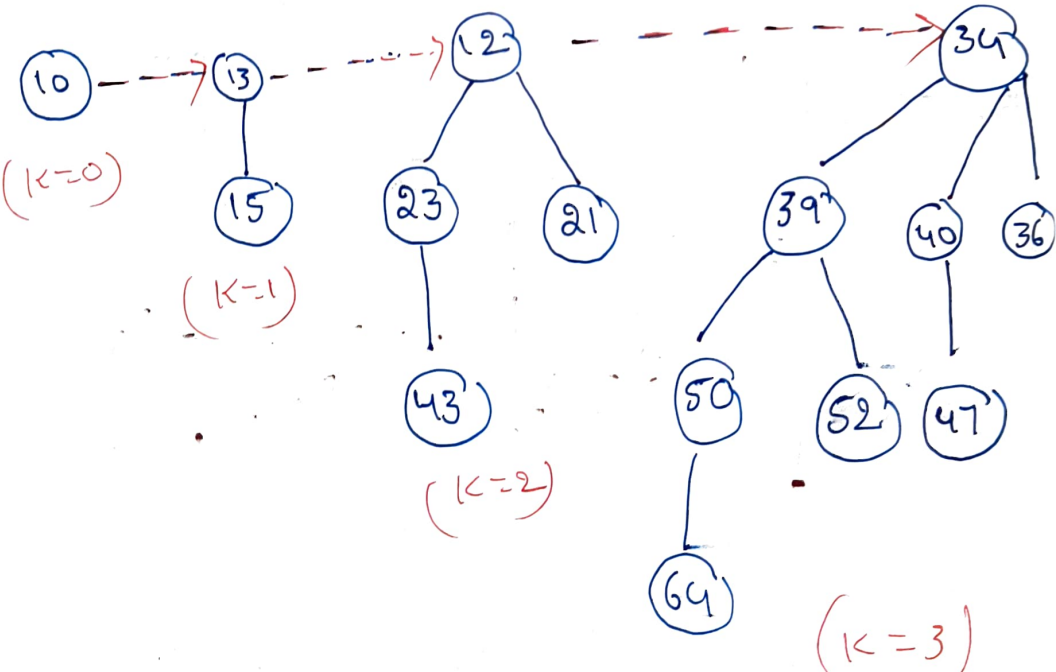
(ii) Binomial Heap :-

Binomial Heap is a collection of Binomial Trees. These binomial Trees are linked together in a specific way.

properties :-

* Each Binomial Tree in the heap follows min-heap property.
* No Two Binomial Trees in the heap can have the same number of nodes.

Ex



(iii) operations * on * Binomial Heap :—

(1) Insert

(2) Merge (or) Union

(3) Extract-Min

(4) Decrease-key

(5) Delete

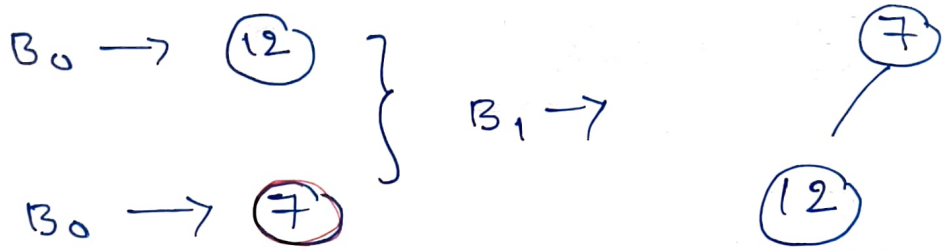
(1) Insert :—

* Each insertion starts as a B_0 Tree.
* At most one Binomial Tree of any degree is allowed.
* If two Trees have the same degree, they are linked.
* while linking:
  - → The Tree with Smaller key as parent.
  - → The other Tree becomes it's Left child.
* Tree degrees increase by 1 after link.
* The Heap always maintain Min-heap property.

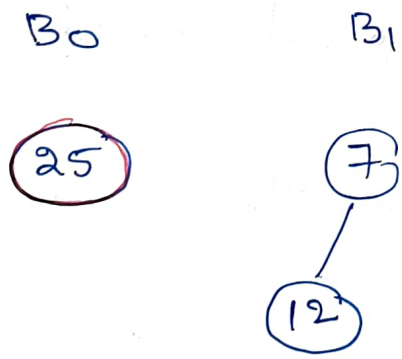
Ex :- [12, 7, 25, 15, 28, 33, 41, 10, 18, 5]

step 1: insert 12

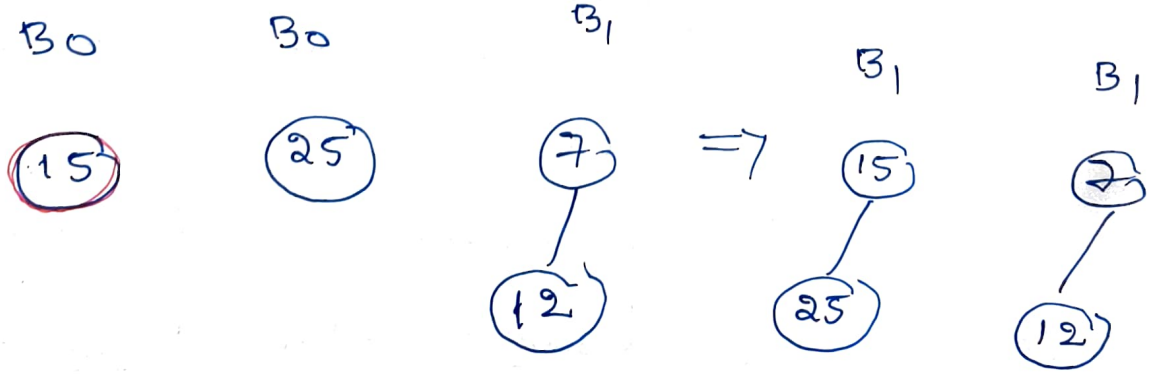
step 2: insert 7



step 3 : insert 25

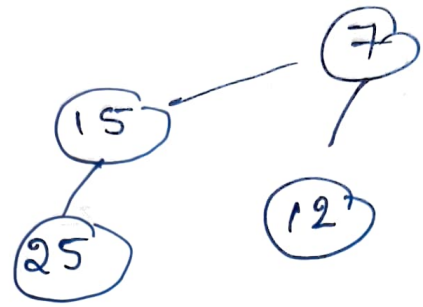


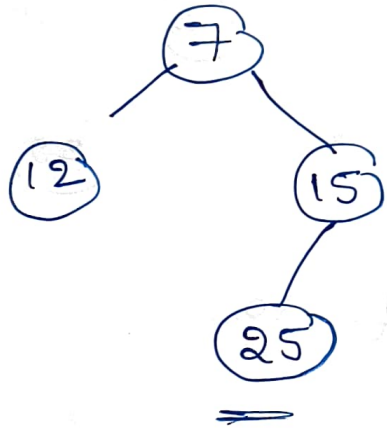
step 4 : insert 15



$B_1 + B_1 = B_2$

$B_2 \rightarrow$

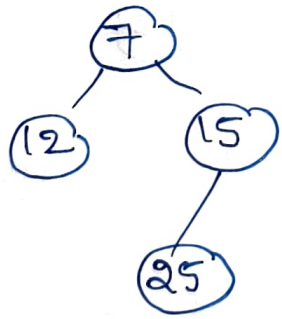




step-5 insert 28

B_0

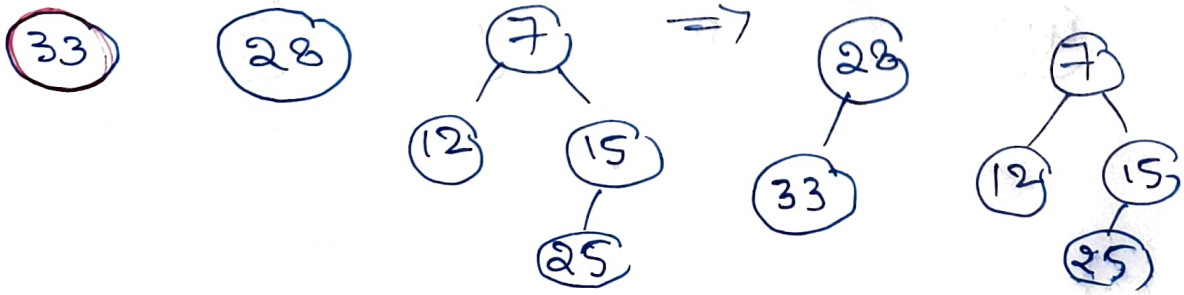
B_1



[degree not same so no link]

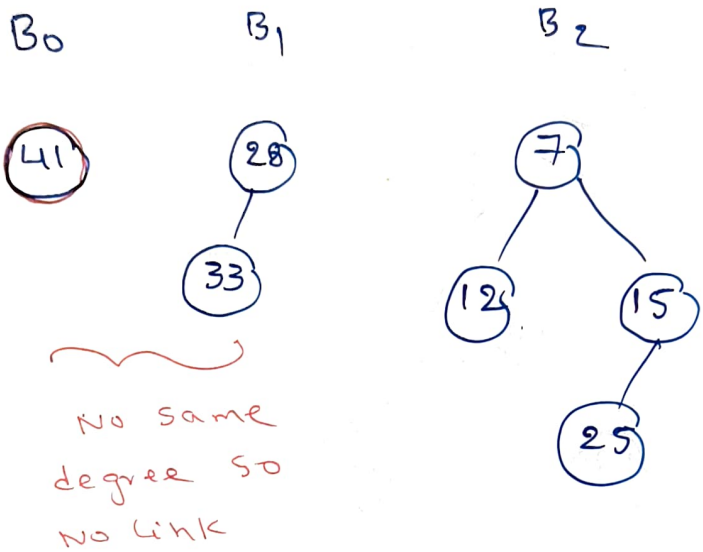
step-6 insert 33

B_0 B_0 B_1

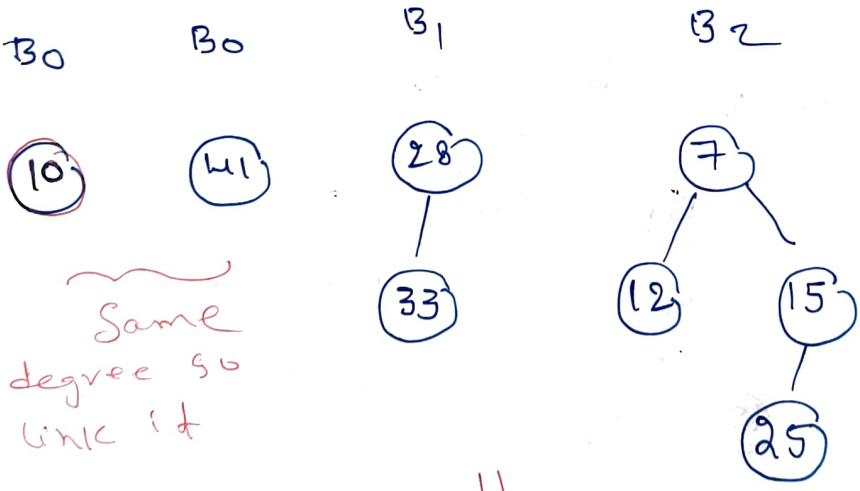


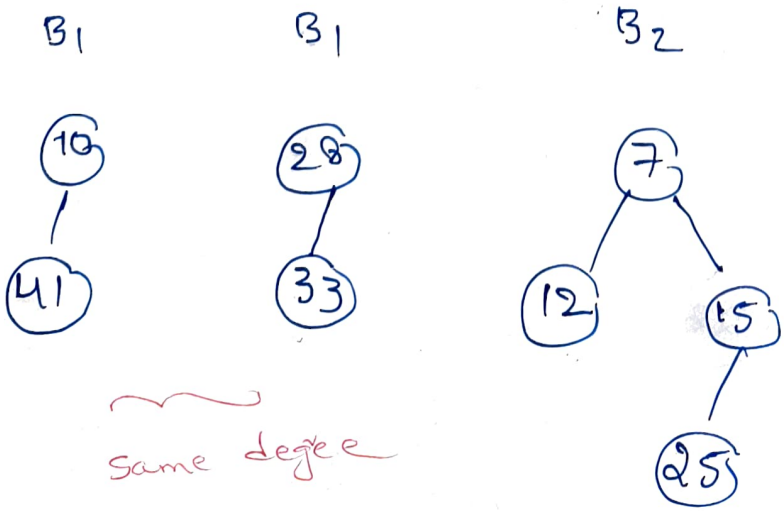
NOTE Here c Min-heap property used

Step-7:- insert 41

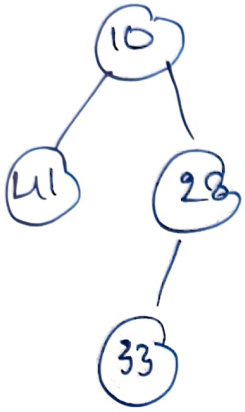


step -8 :- insert 10

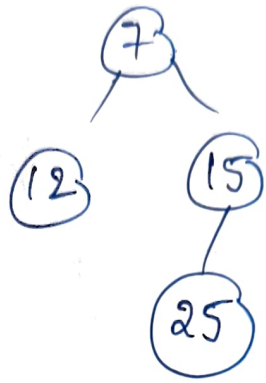




B_2

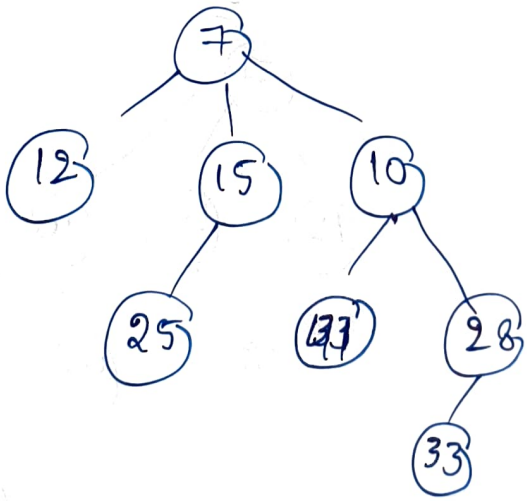


B_2



$B_2 + B_2 = B_3$

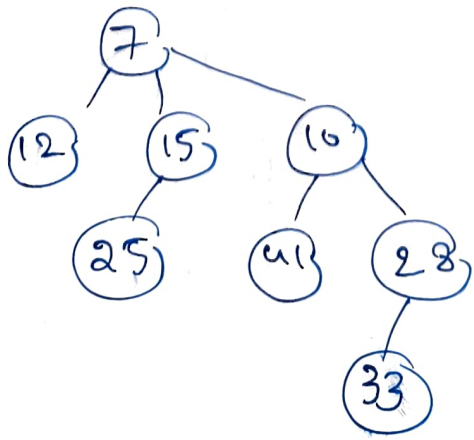
$B_3 = 7$



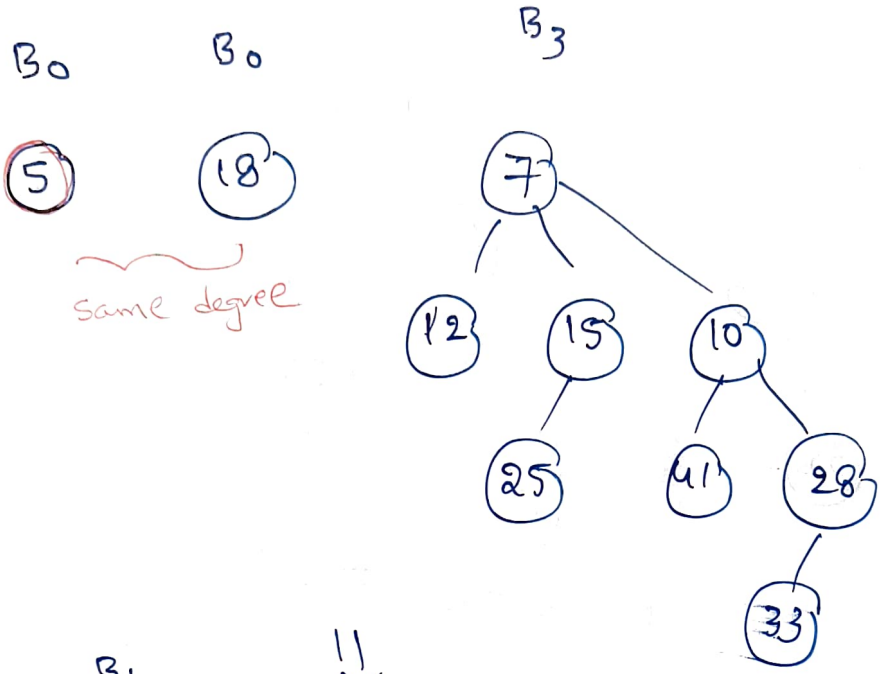
Step :- 9 insert 18

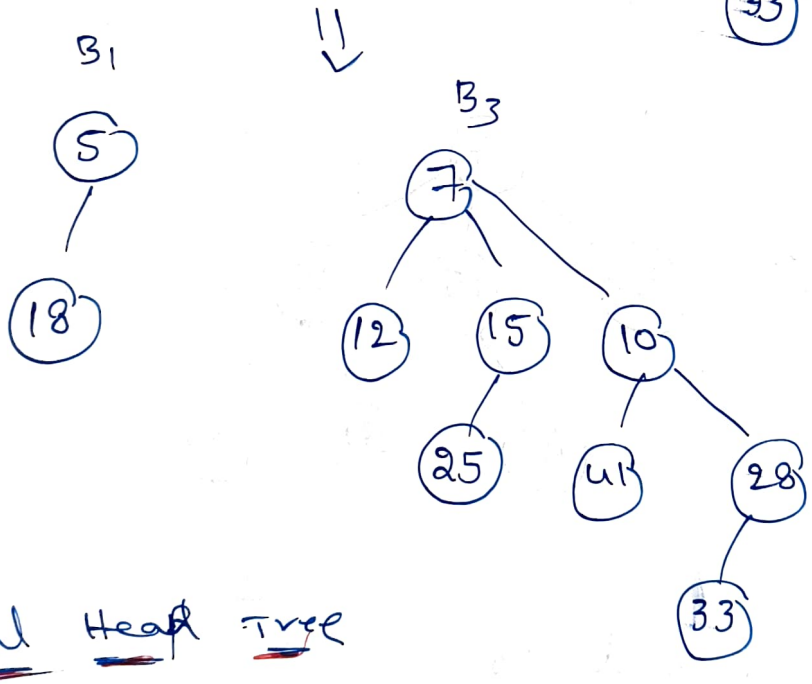
B_0

B_3

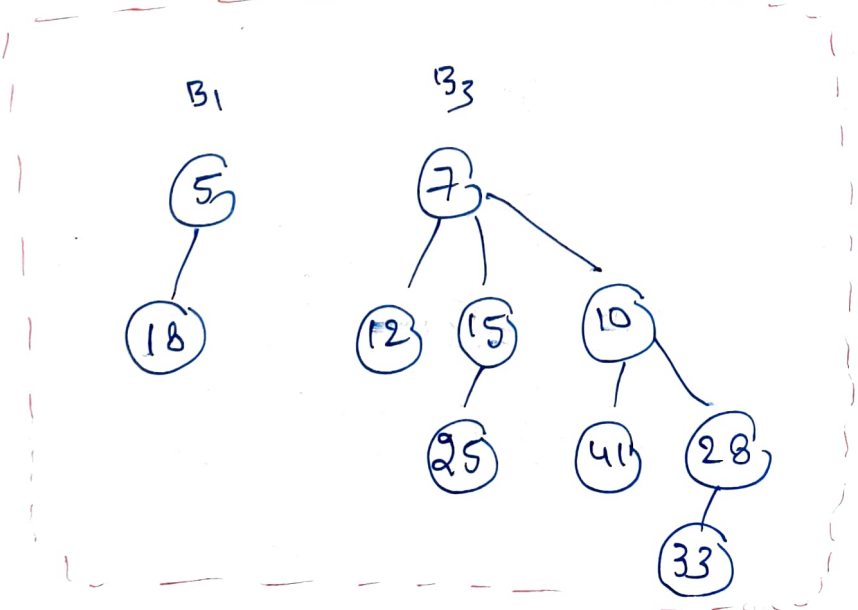


Step 10: insert 5





Final Heap Tree

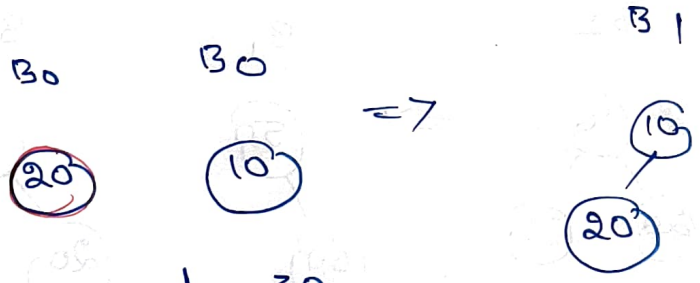


Ex-2:- Insert 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120 in Binomial Heap

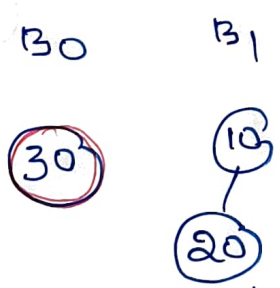
solution:-

step 1:- insert 10

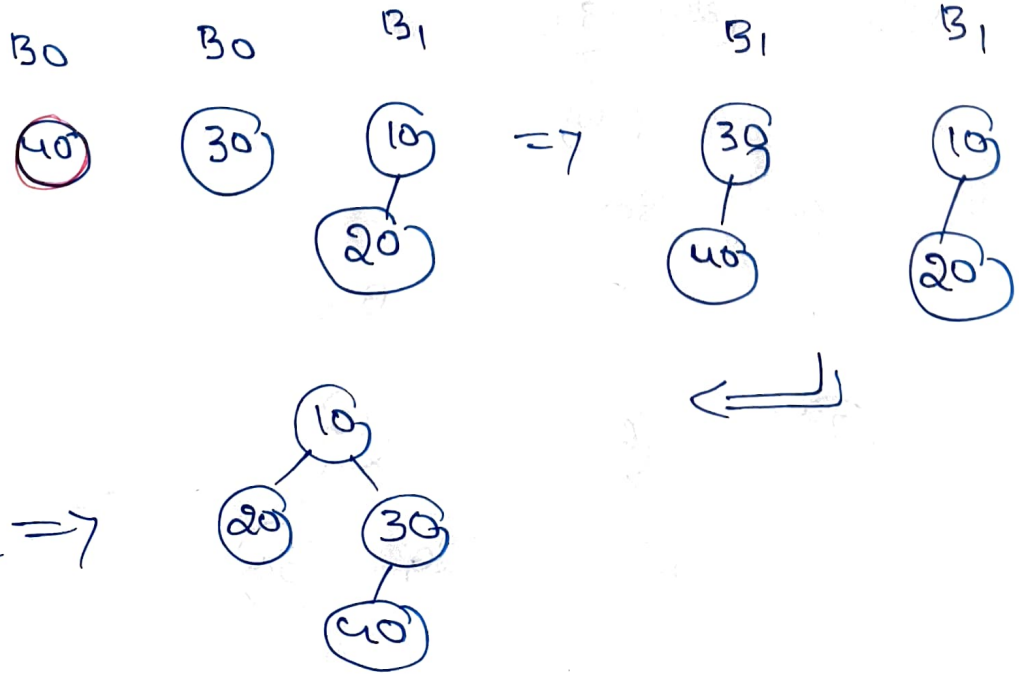
step 2:- insert 20



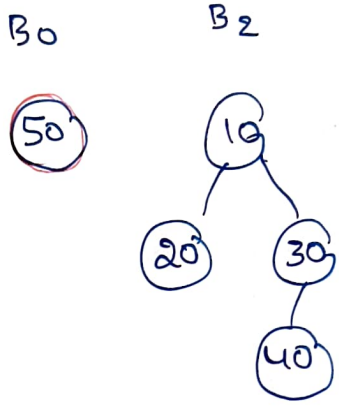
step 3:- insert 30



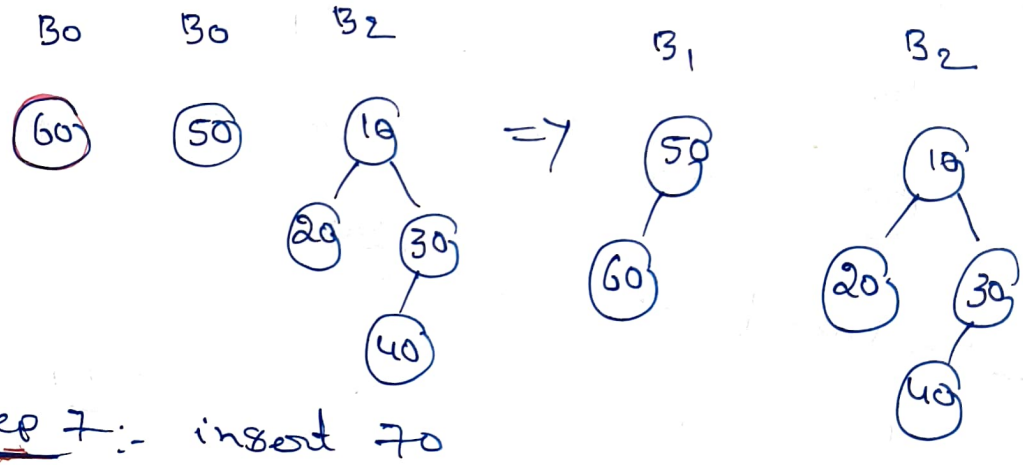
step 4:- insert 40



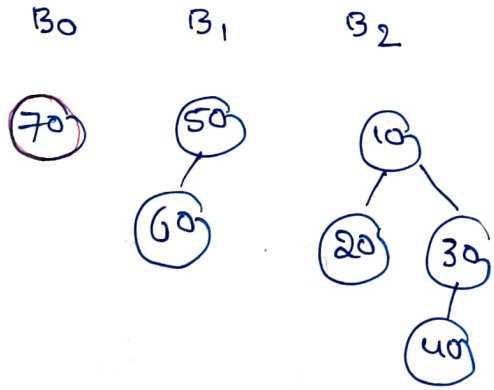
Step 5:- insert 50



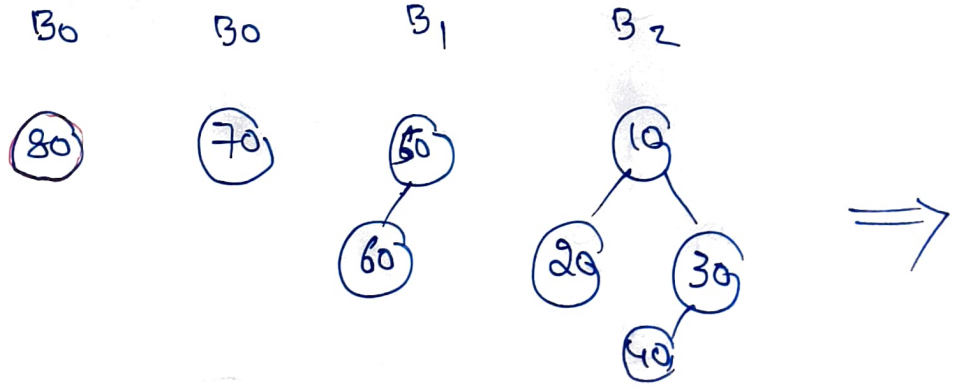
step 6: insert 60



Step 7:- insert 70



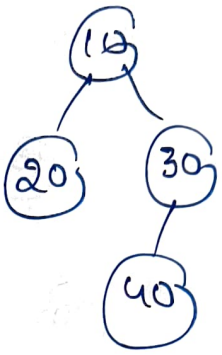
step 8 insert 80



B_1

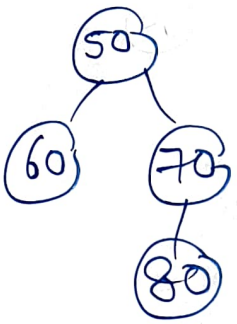
B_1

B_2

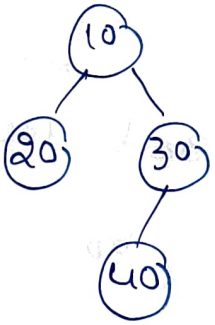


$B_1 + B_1 = B_2$ ||

B_2



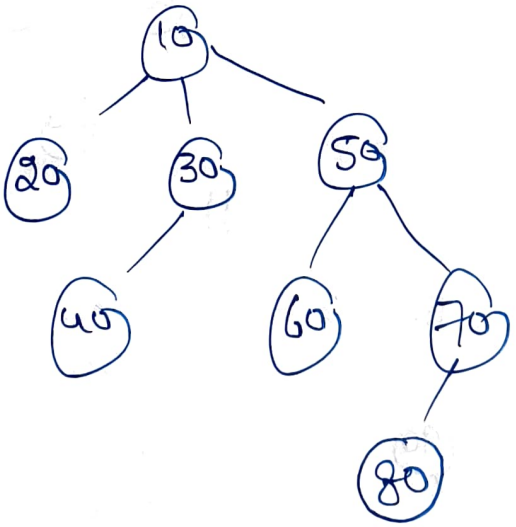
B_2



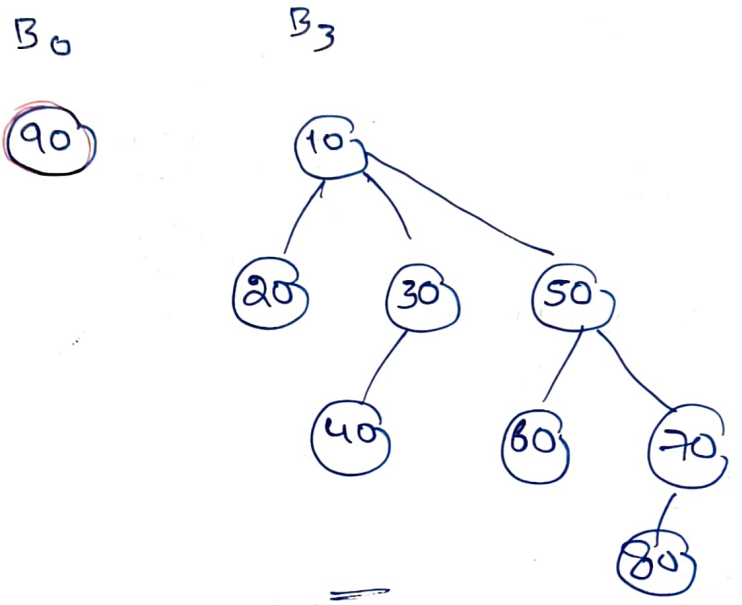
||

$B_2 + B_2 = B_3$

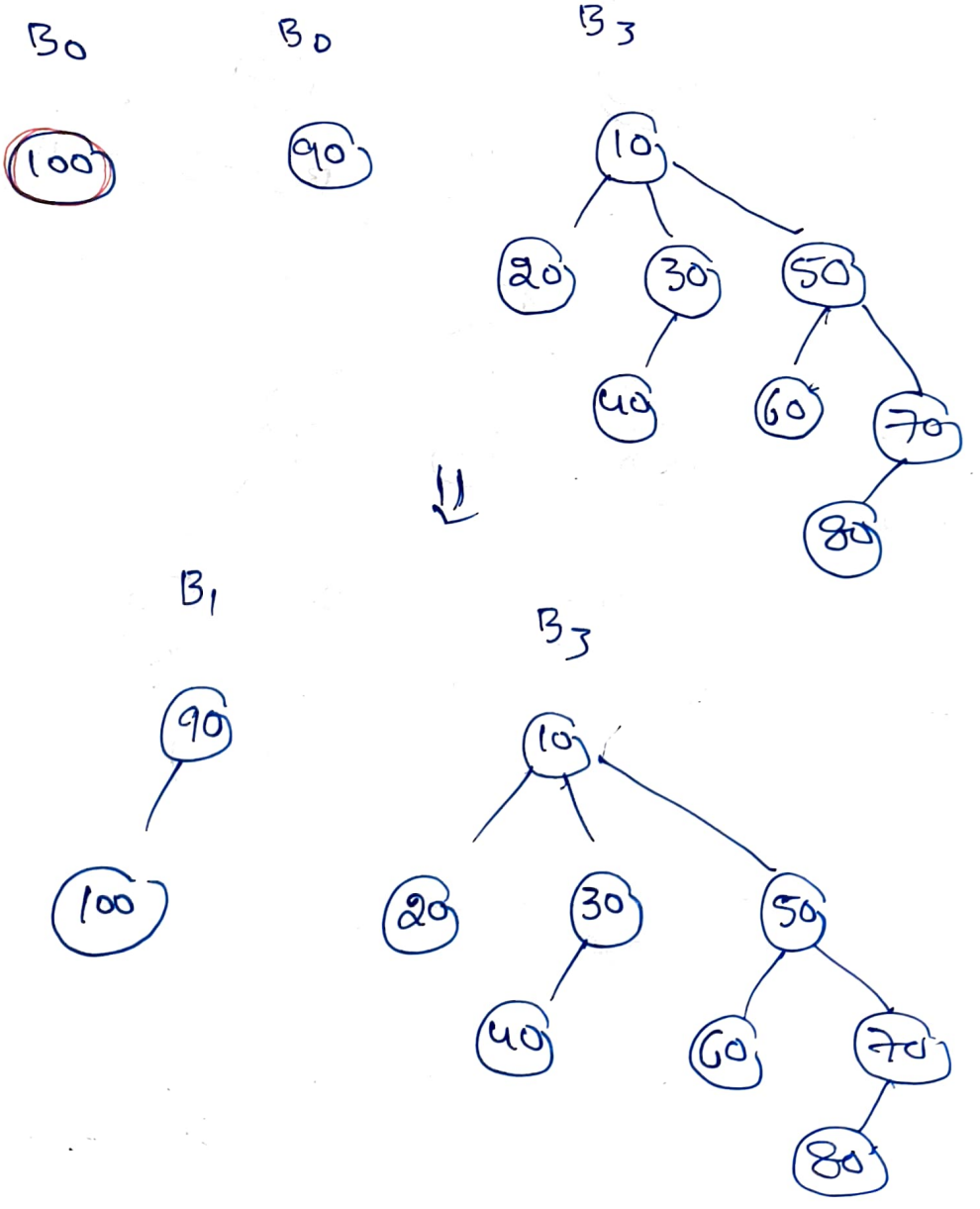
$B_3 =$



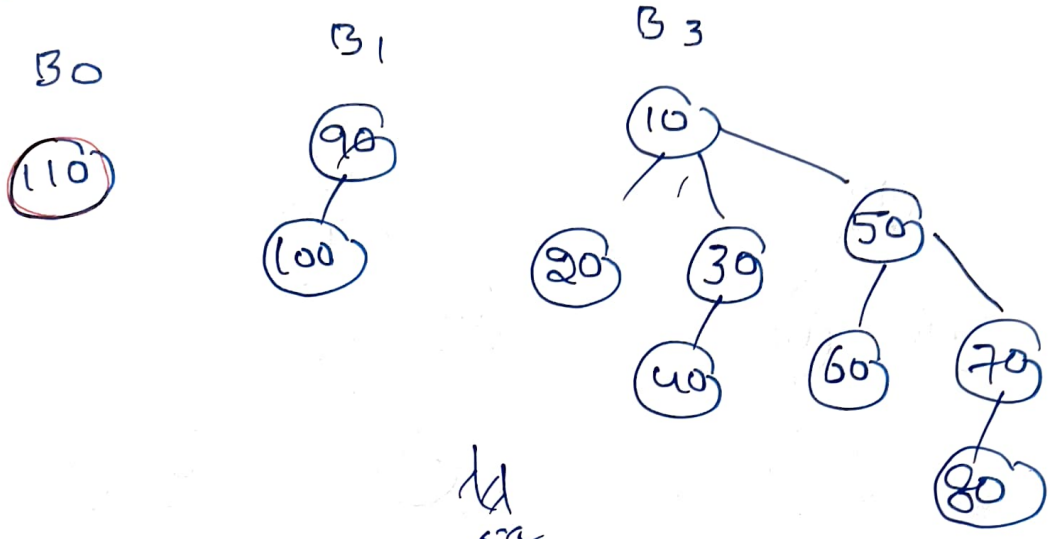
step 9 :— insert 90



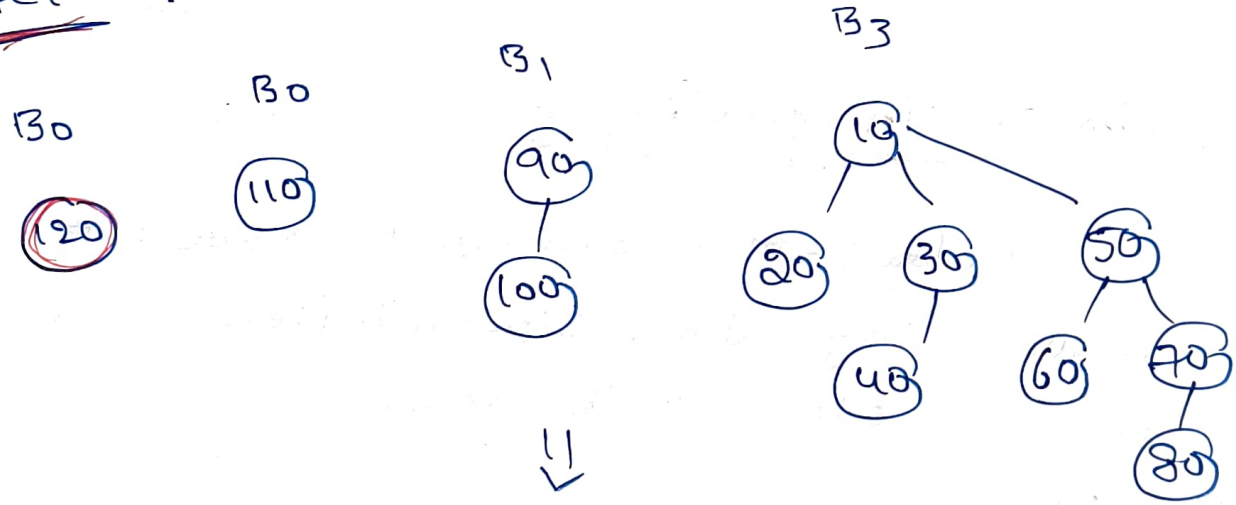
step 10 : insert 100

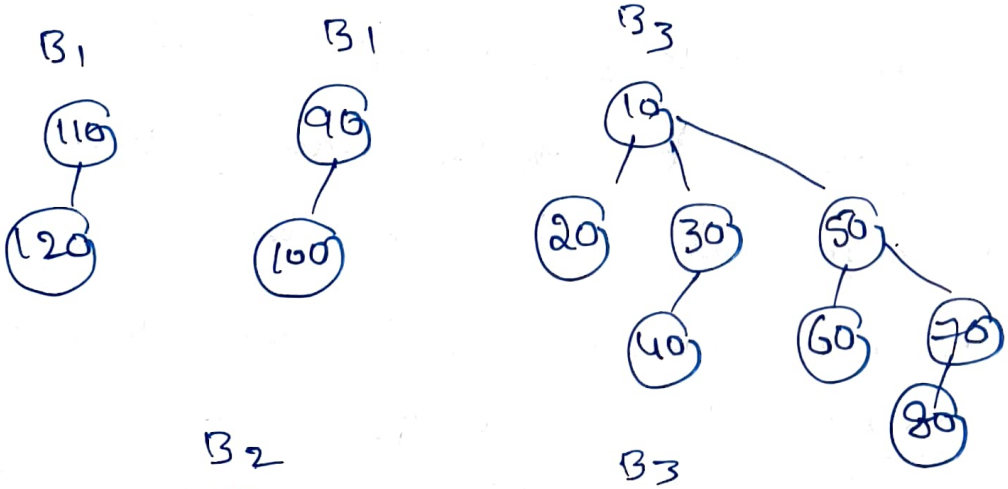


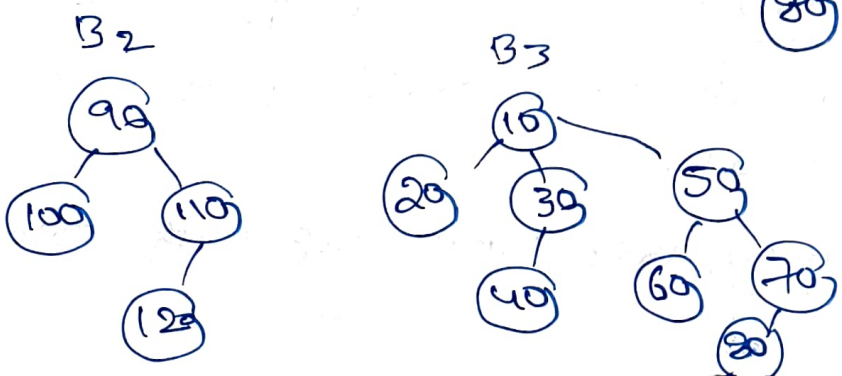
Step 11 :- insert 110



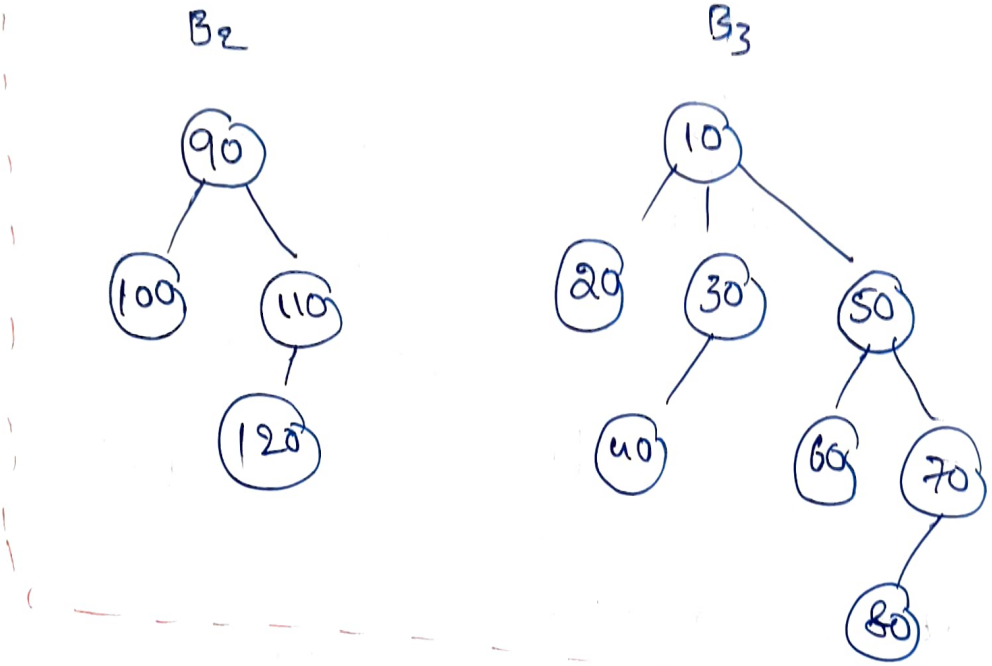
Step 12: insert 120







Final Heap



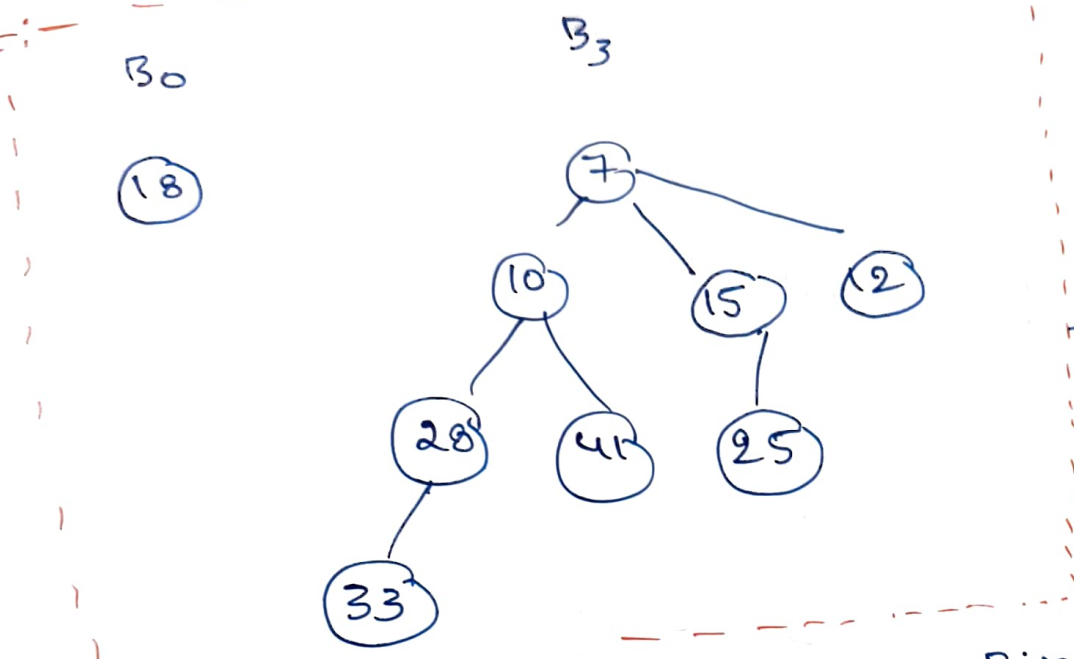
2 Extract-Min operation :-

Extract-Min removes the node with smallest key from Binomial Heap and then re-structure the heap.

Rules :-

* The minimum key is always the root
* Find the root with the smallest key
* Remove that root from the Heap.
* Take the children of the removed root, reverse their order.
* Treat these children as a new Binomial heap.
* Union this new heap with remaining heap.

Ex:-



Binomial heap.

Apply Extract-Min to above Binomial heap.

step 1:- Find minimum Root.

Min (18, 7) = 7.

min = 7

step 2:- Remove the Root 7 then Take the children of the removed root & reverse their order i.e

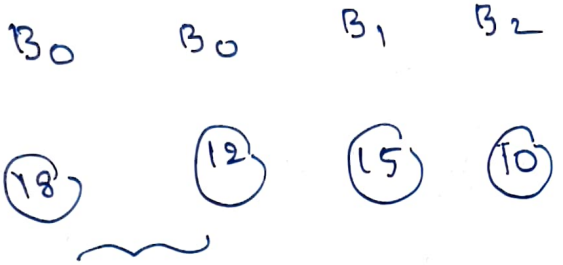
10 15 12 → 12 15 10

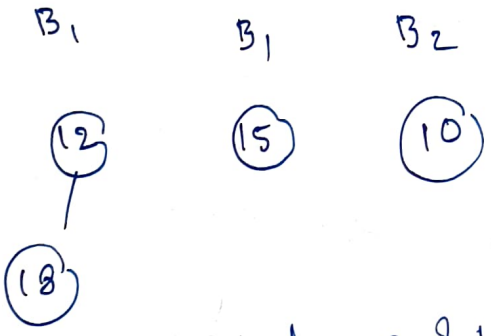
Now New Binomial Trees.

B0 B1 B2

12 15 10

Union with remaining heap.

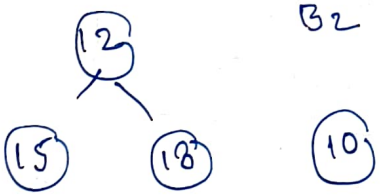




Now B_1 conflict: 12 & 15

smallest root = 12

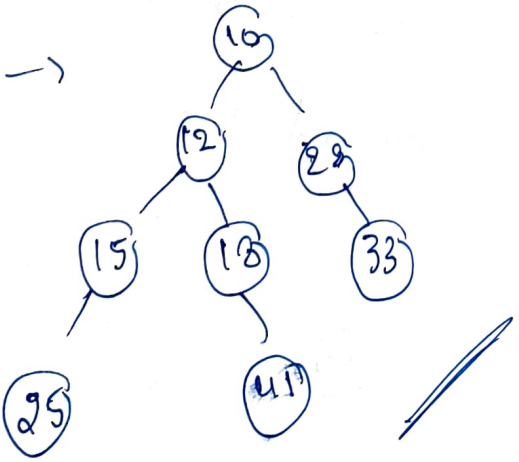
B_2:



Now B_2 conflict: 12 & 10

smallest root = 10

B_3 →



3 Decrease key :—

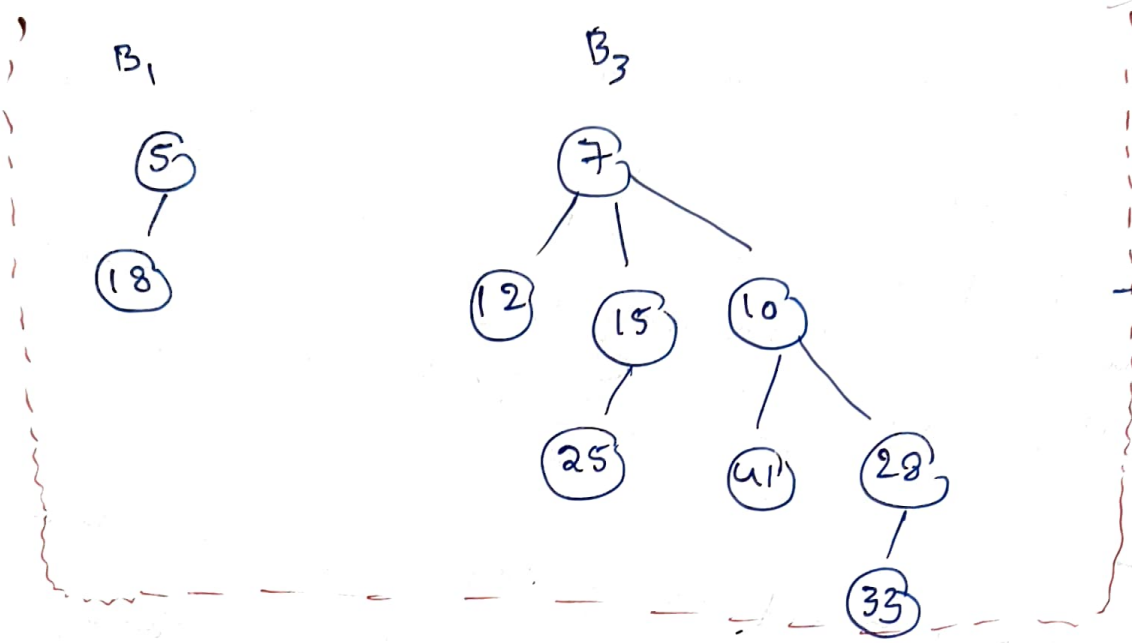
The Decrease key operation reduces the value (key) of a given node in a Binomial heap. After Decrease the key, the Heap is restructured by moving the node upward until the min-heap property is restored.

Rules :—

* The new key must be smaller than the current key
* Replace the node key with new (smaller) key.
* if the heap property is violated (node < parent)
  * swap the node with it's parent.
  * continue swapping upward (bubble-up) until
    * The node becomes a root or
    * The parent key is smaller

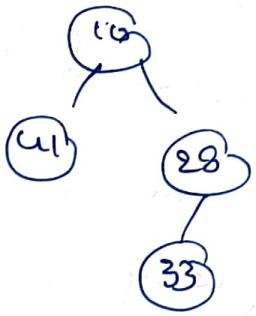
* The structure of the binomial trees does not change, only keys are Exchanged.

EX:-

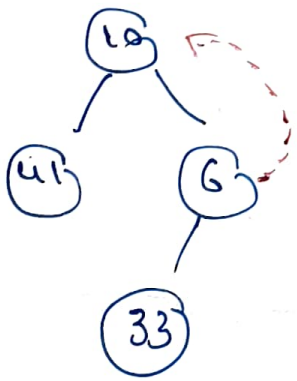


Decrease key node (28) → (6)

Step-1:- Locate node 28.



Step-2 Decrease key value i.e 28 → 6

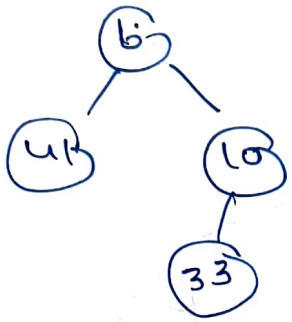


Heap property violated

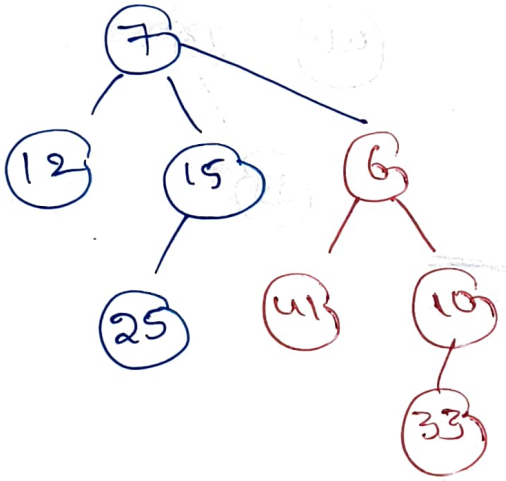
(6 < 10) X

swap to parent

step 3:- 6 → 10



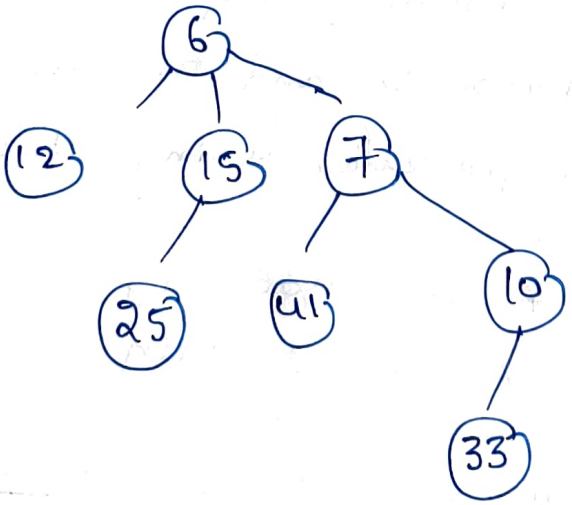
step 4:-



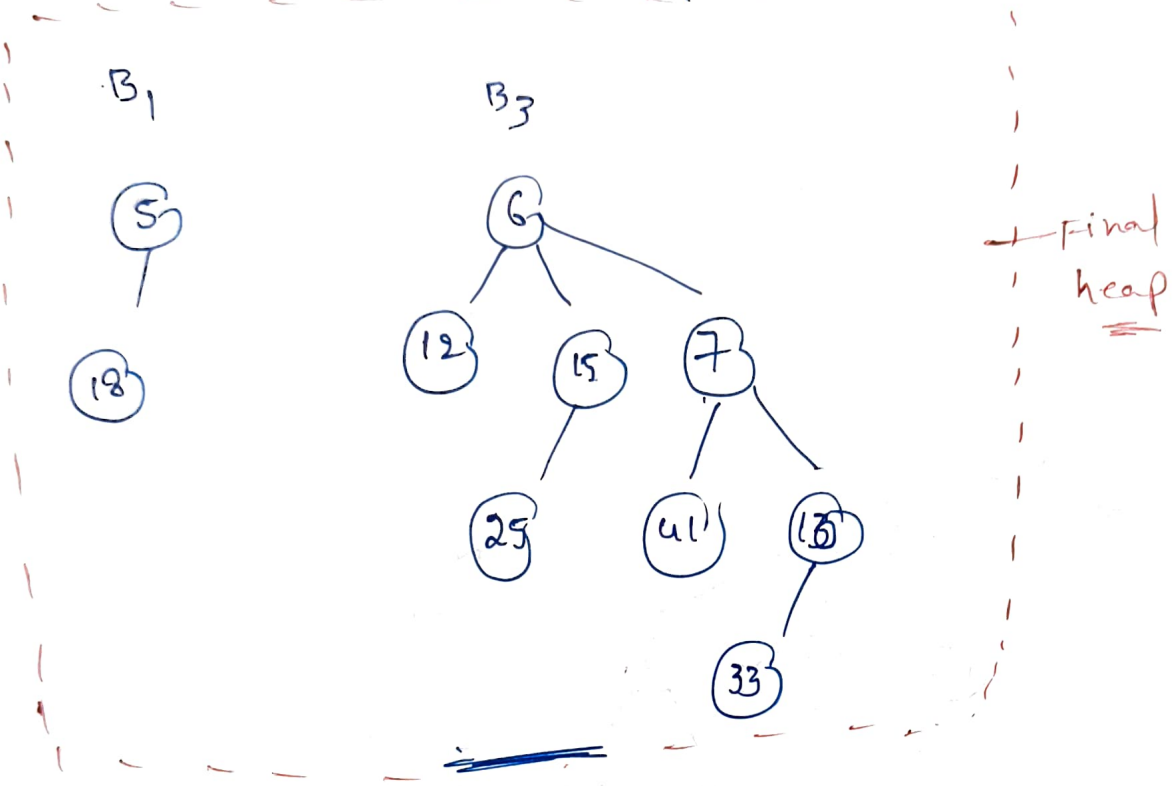
Now compare with new parent 7

(6 < 7) ✗

swap to parent.



Steps Final heap after Decrease-key.



4. Delete :-

Delete operation removes a specific node from a Binomial heap and maintain min-heap property and Binomial heap structure.

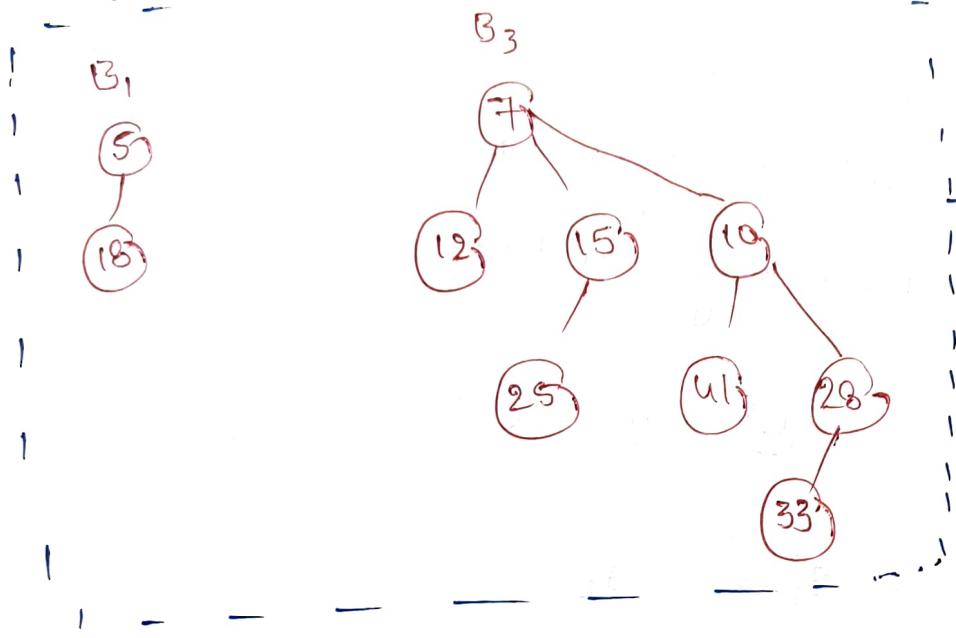
→ Delete is implemented using two Existing operations.

* Decrease-key
* Extract-Min

Rules :-

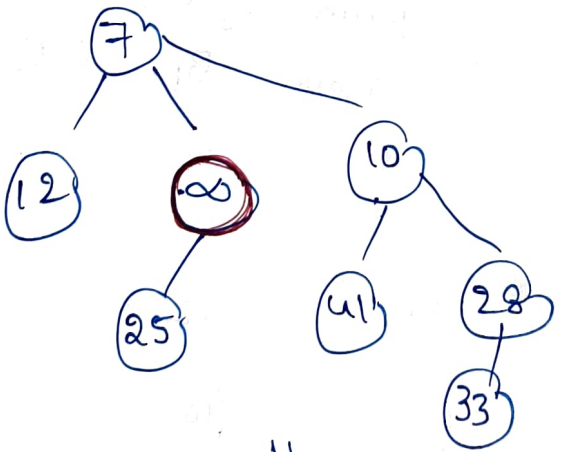
* Decrease the key of node x to $-\infty$
* This causes x to bubble up to the root.
* perform Extract-Min to remove it
* Union the remaining trees to restore the heap

Ex:- [12, 7, 25, 15, 28, 33, 41, 10, 18, 5]



Sol:- Delete Node 15

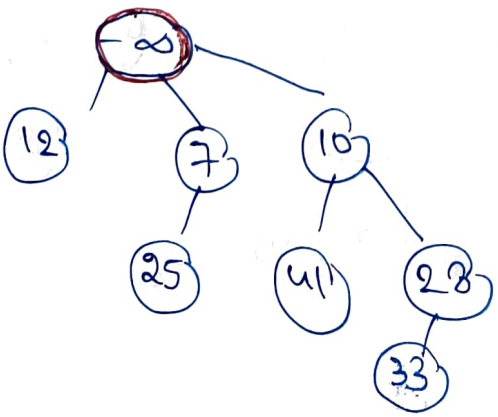
Step-1:- Decrease-key of node 15 to $-\infty$



$(-\infty < 7)$

Swap to parent

$-\infty \rightarrow 7$



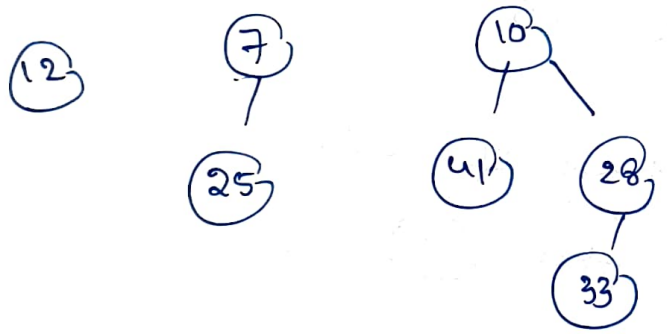
Node reached to root then stop

Step 3:- Extract-min (Remove $-\infty$)

→ Remove the root $-\infty$ mean delete original node 15.

→ Remaining children of $-\infty$

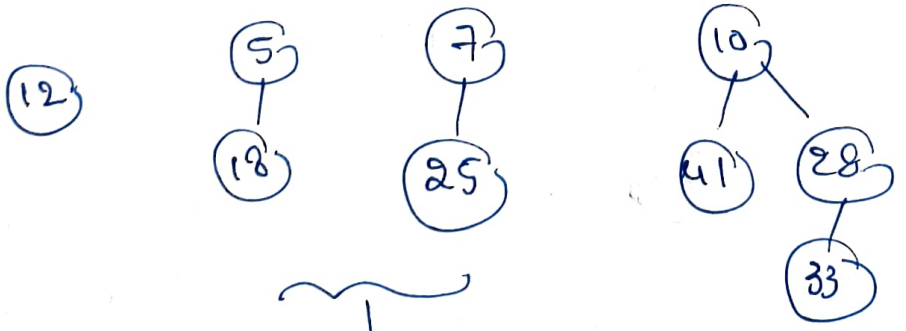
B_0 B_1 B_2



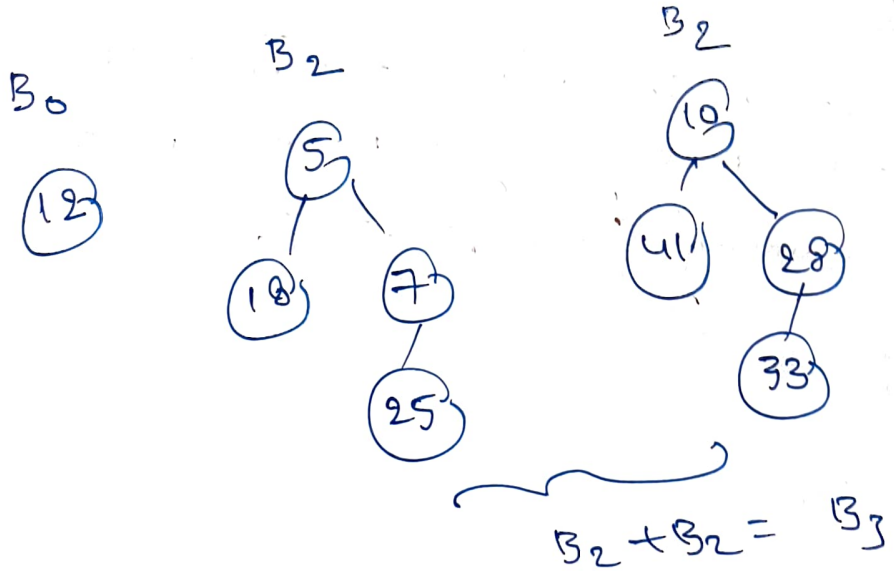
From given question. we have B_1. i.e

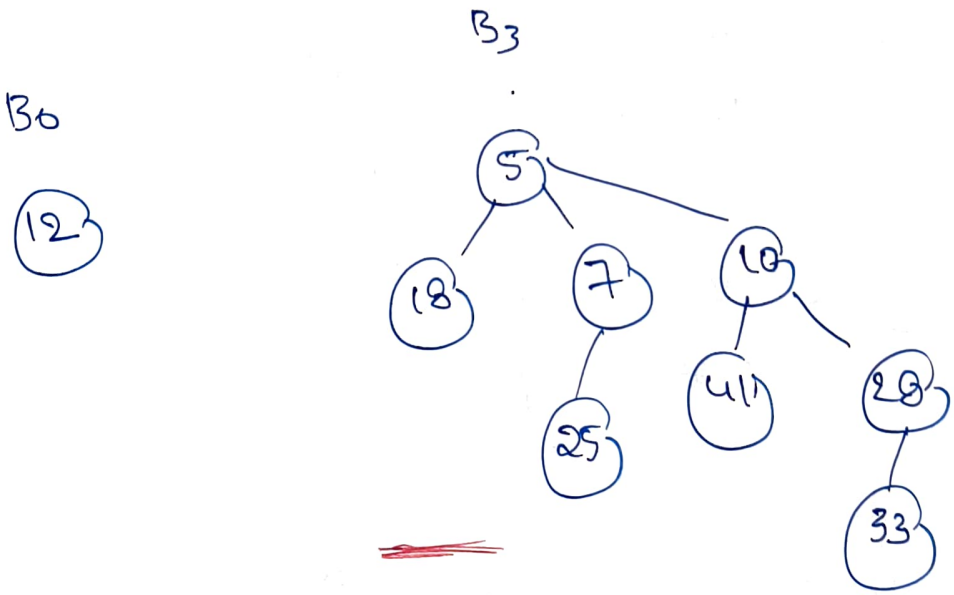
B_1

B_0 B_1 B_1 B_2



$B_1 + B_1 = B_2$





NOTE

→ Delete does not directly remove a node.

→ It uses Decrease-key + Extract-min

→ Tree structure is preserved.

→ Runs in $O(\log n)$ Time

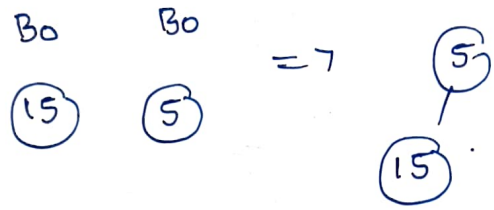
problem - 1

Demonstrate the delete operation in Binomial heap by deleting key 15 from a heap containing {5, 15, 25, 35}.

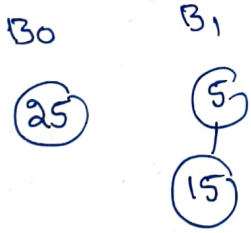
Solution :-

step 1 :- B_0

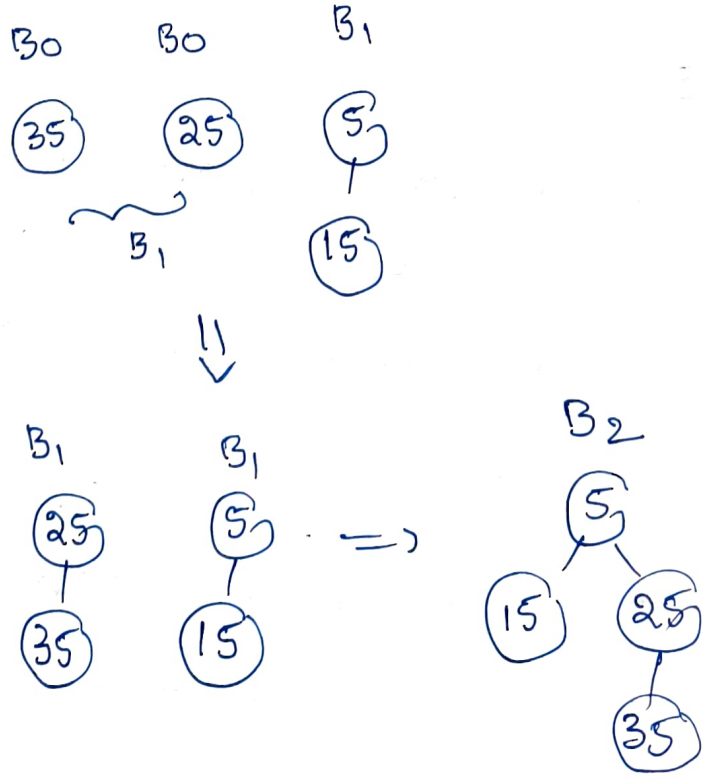
step 2 :-



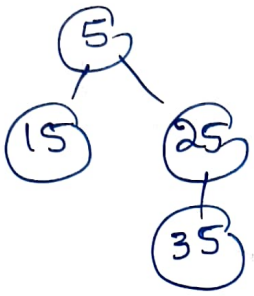
step 3 :-



step 4 :-



After inserting Binomial heap is.

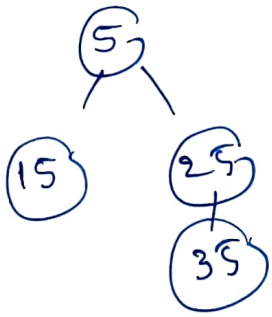


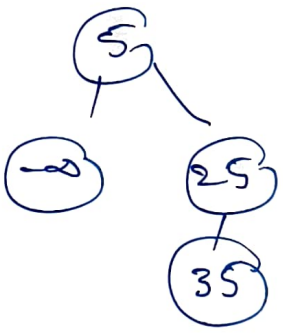
Now Delete-key = 15.

→ Delete operation is performed in two phases.

* Decrease-key (15 → -∞)
* Extract-min ()

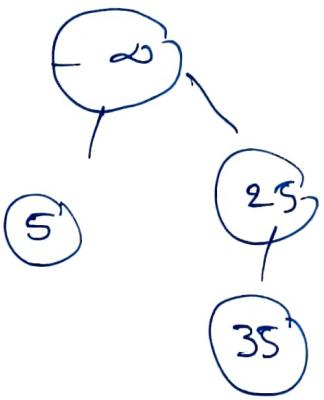
Step-1 Decrease (15 → -∞) i.e -15 → -∞





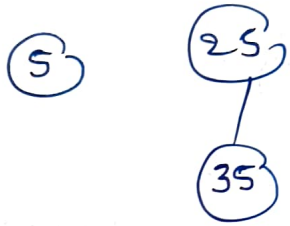
-∞ < 5

swap to parent.



step-2 :- Extract-min (Remove -5).

children of deleted node



Final heap after removing 15