Unit V

Advanced Data

(1) Fibonacci heaps

A fibonacci heap is a collection of rooted trees that are ~~mign~~ heap ordered

Fibonacci heap data structure serves dual purpose.

* First, it supports a set of operations that Constitutes what is known as a "mergeable heap",
* Second, several fibonacci - heap operations run in constant ~~tim~~ amortized times which make this data structure well suited for applications that invoke these operations frequently.

Mergeable heap is any data structure that supports the following five operations, in which each element has a key:

(1) Make-Heap() Creates & ~~retrn~~ returns a new heap containing no elements

(2) Insert (H, x) inserts element x, whose key has already been filled in H

Minimum (H) → returns a pointer to the element in heap 'H' whose key is minimum.

Extract Min(H) deletes the element from heap H whose key is minimum, returning a a pointer to the element.

union (H_1, H_2) – Creates & returns a new heap that contains all the elements of heaps H1 & H2.

In addition to the mergeable operations above, Fibonacci heaps also support the following two operations:

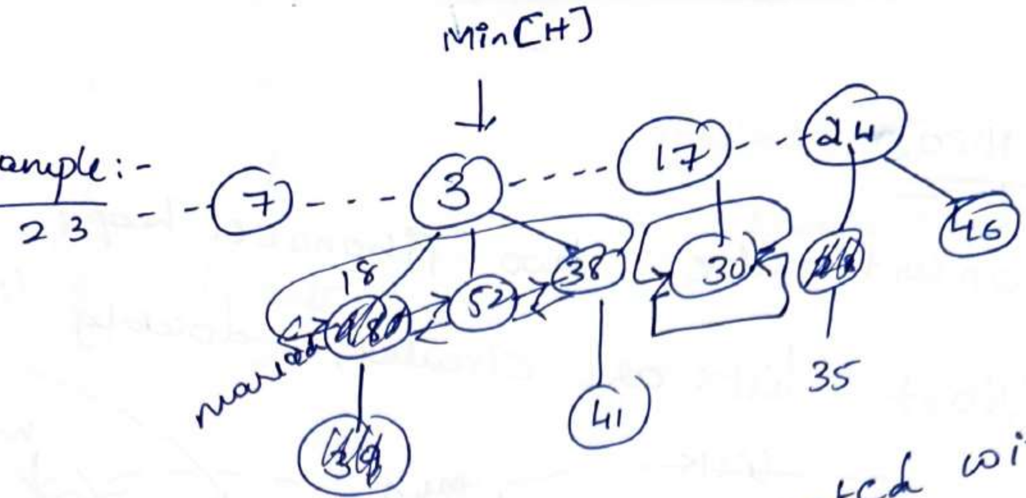
Decrease KEy (H, x, K) assigns to element x, within heap H the new key value k, which we assume to be no greater than its current key value.

Delete (H, x) deletes element x from heap H.

Characteristics of Tree 'wi'

1. Trees are ~~not~~ necessarily bionomial
2. Siblings are Bi-directionally linked
3. ,

Example:-



The root nodes are connected with circular doubly linked list

Siblings are connected through circular doubly linked list.

Fibonacci heap operations

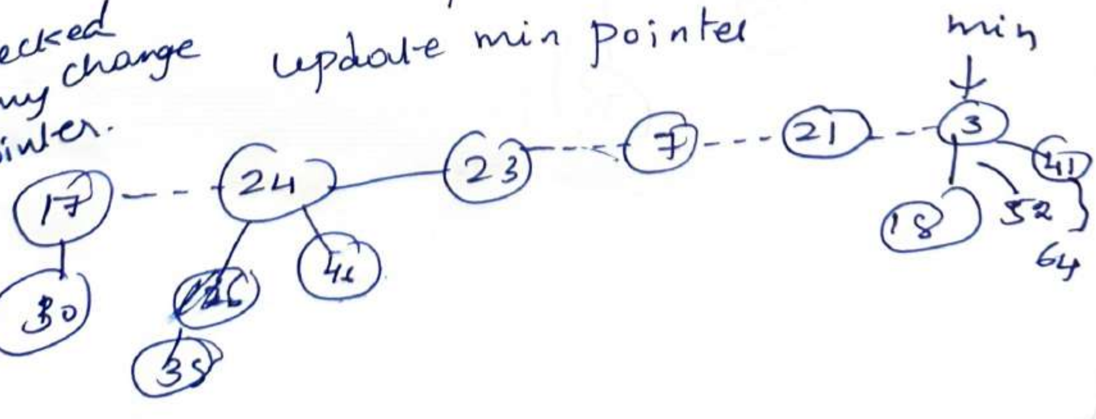
① Creation, Insertion, Finding min key, union, Extract minimum key, Decrease key, Deletion.

~~Creation~~ Fibonacci Heaps: Insert

Create a new singleton tree

Add to left to min pointer.

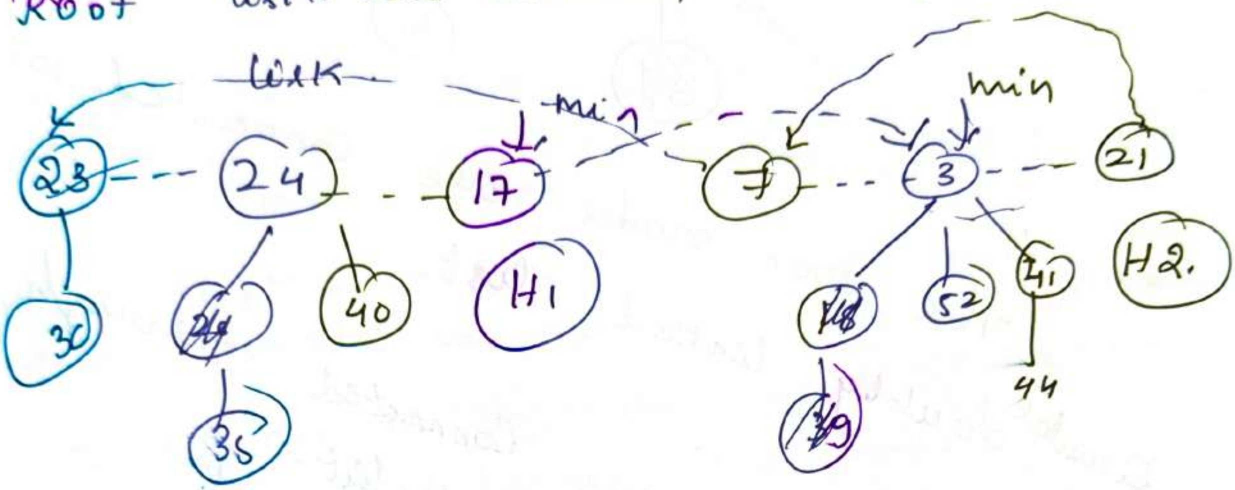
Then it has to be checked if any change update min pointer in min pointer.



Union

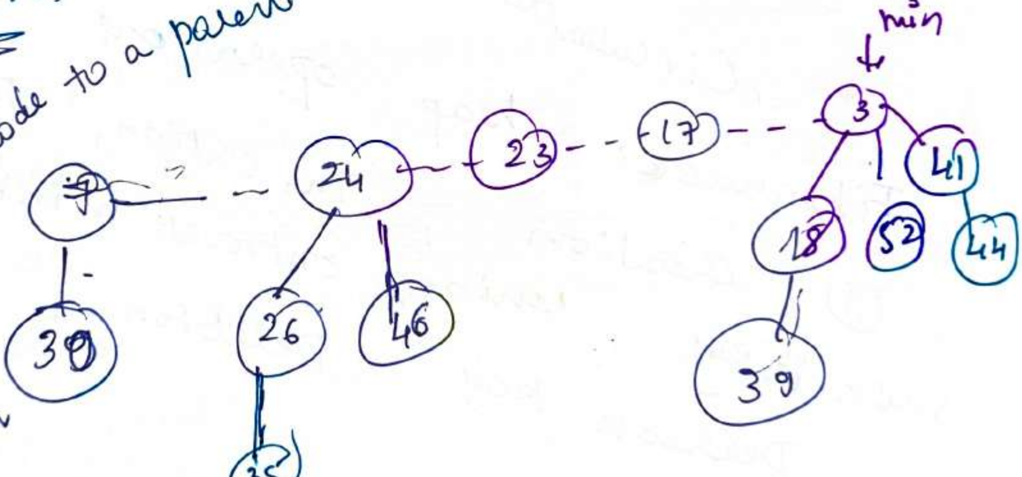
concatenate two Fibonacci heaps

Root lists are circular, doubly linked

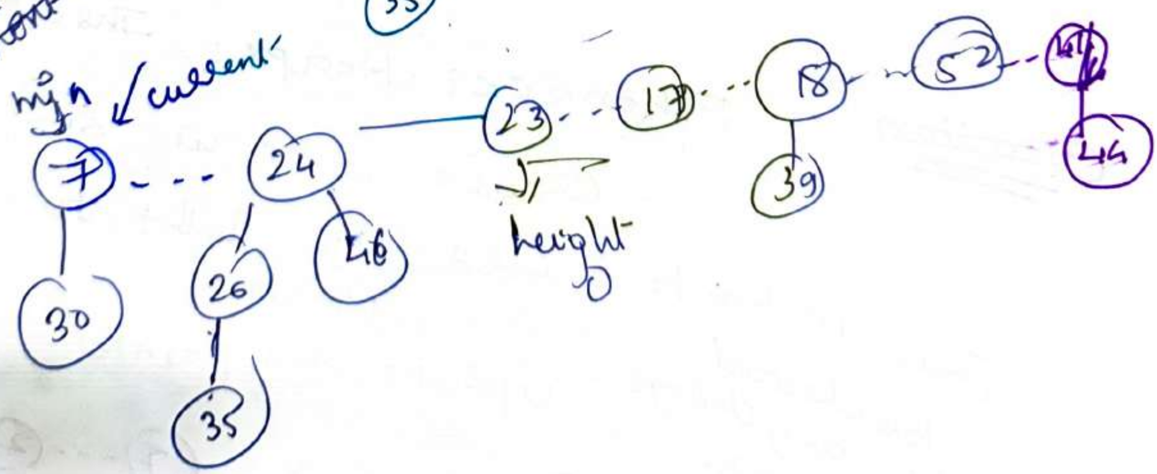


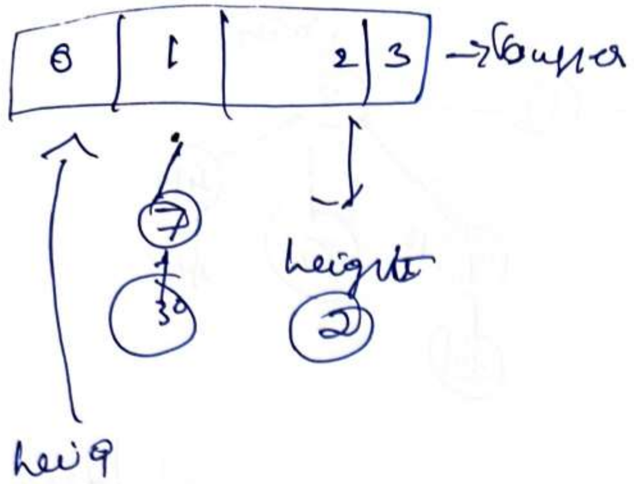
Delete/Min (Extract-Min.

Degree of a tree → No. of childnode to a parent



Right pointer will be the ~~next~~ min

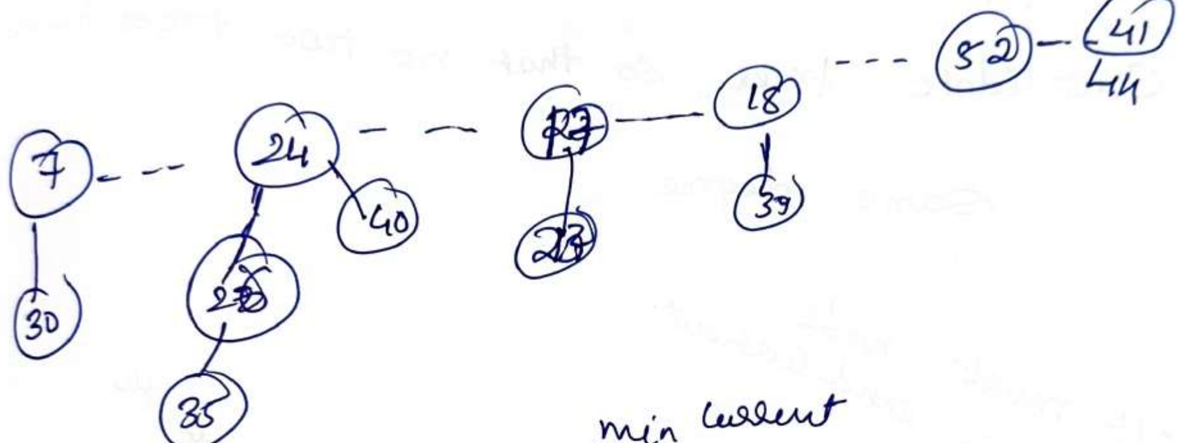


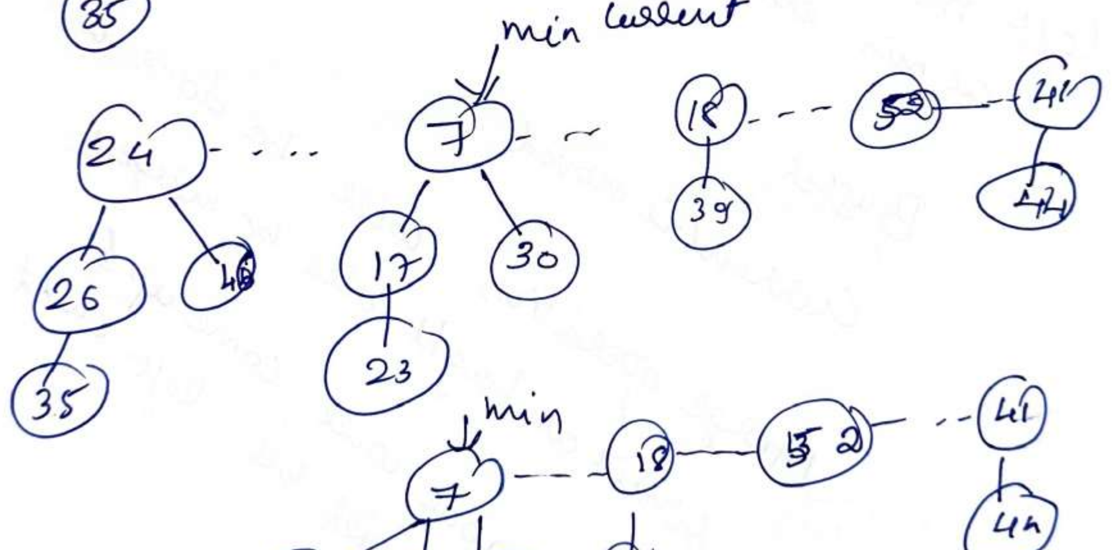


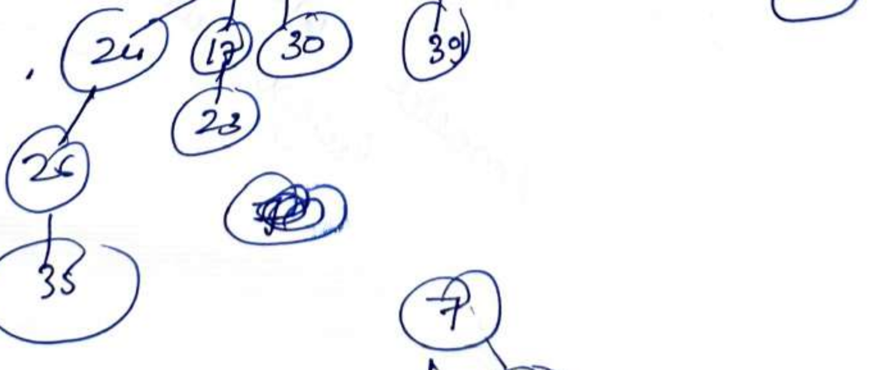
maximum height+1

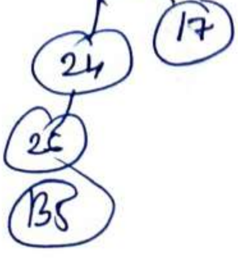
buffer is created

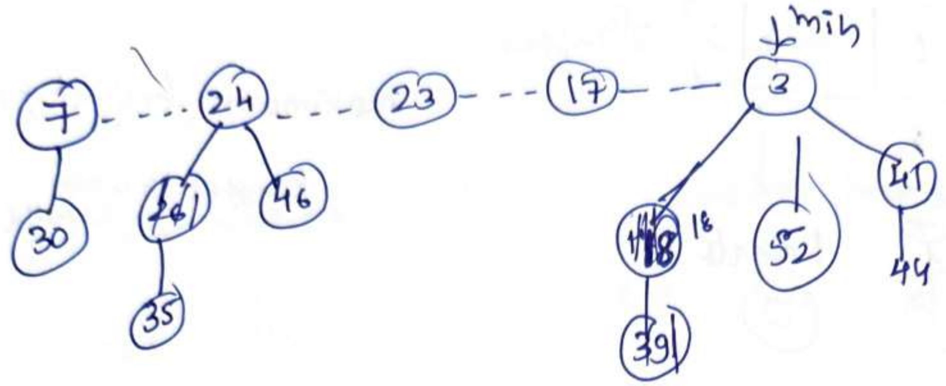
23 & 17 is compared & merged











Delete min & Concatenate its children Into root list.

Consolidate trees so that no two trees have same degree

left most node as min and current.

Buffer

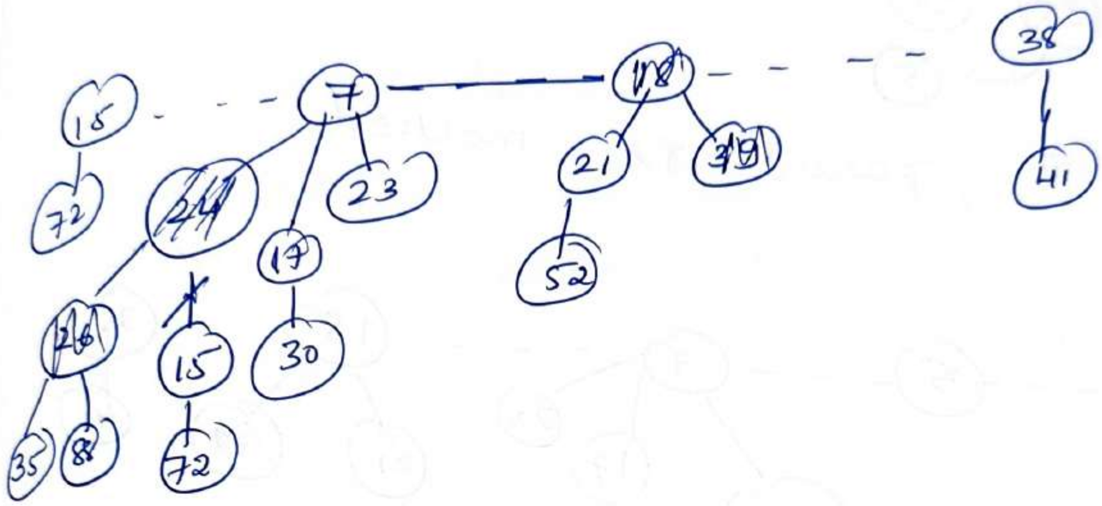
Current i.e moved

Merge operation will be done with similar height will be merged.

smaller node will come up & larger will be left child.

45 to 15

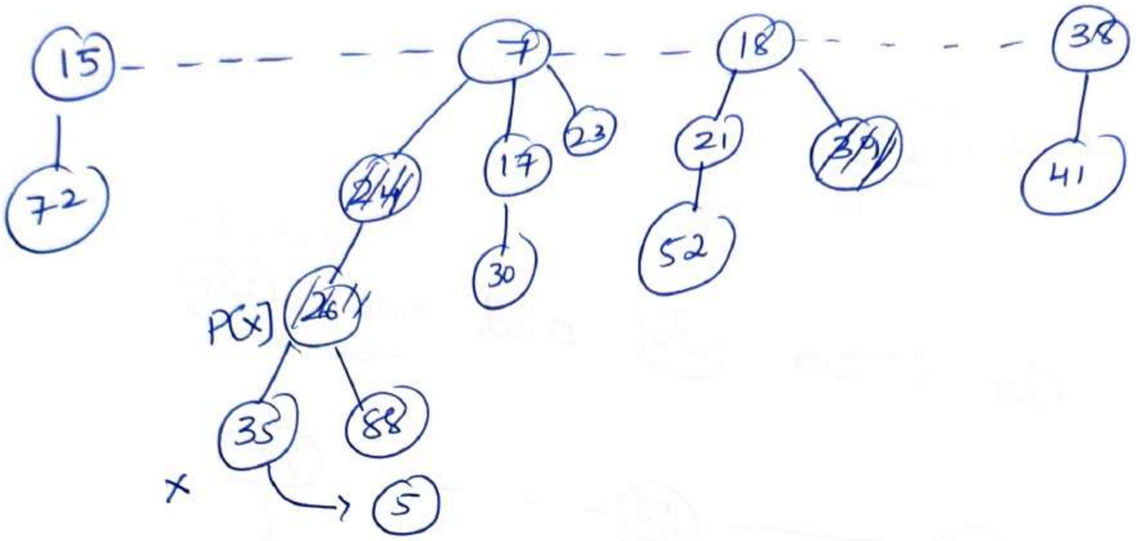
Cut from 24 and mark parent 24.



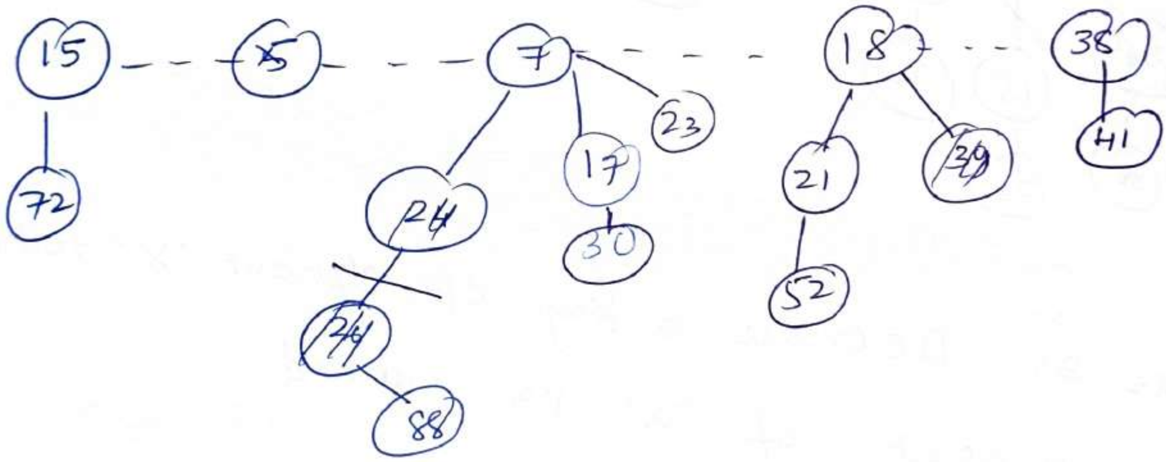
Case 3:· Decrease key of element 'x' to k

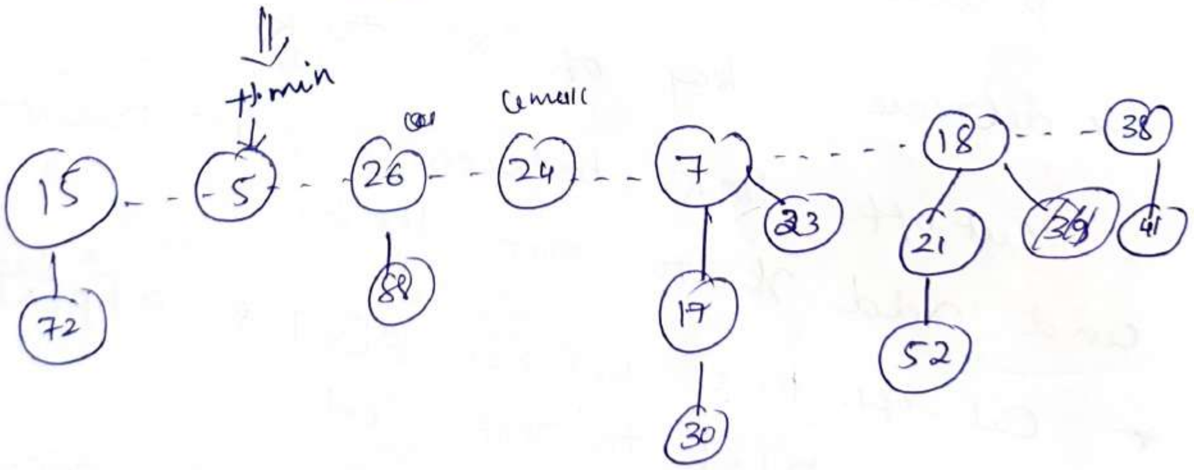
parent of 'x' is marked

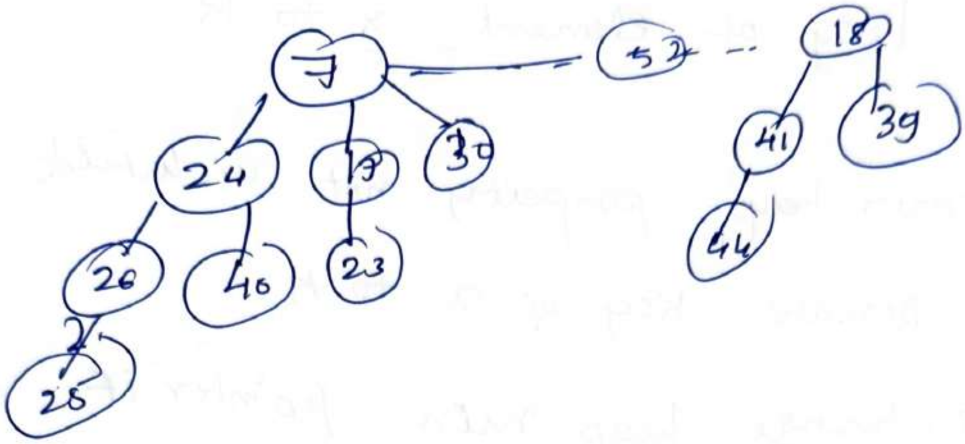
* decrease key of 'x' to k.
* Cut off link between x & its parent p[x] and add x to root list.
* Cut-off link between p[x] & p[p[x]], add p[x] to root list.
  → if p[p[x]], unmarked, then mark it
  → If p[p[x]] marked, cut off p[p[x]], unmark, & repeat.



35 to 5 , parent is marked



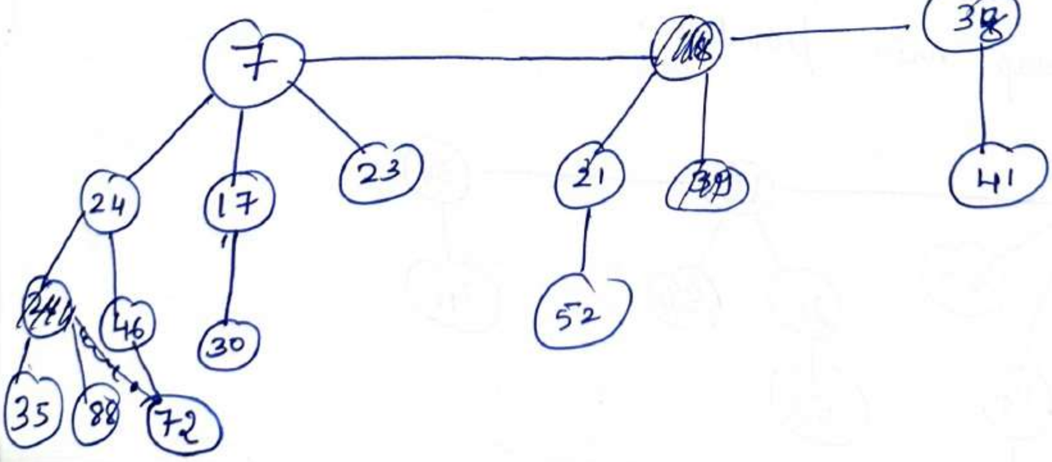




Creating a new fibonacci heap:-

To make an Empty fibonacci heap, the Make-FIB-Heap ~~allotal~~ procedure allocates & returns the fibonacci heap object H. where $H.n=0$ & $H.min = NIL$; there are no trees in H : Because $t(H)$

Decreasing a key:-



Deleting a node:-

The following pseudocode deletes a node from an n-node fibonacci heap in $O(D(n))$ amortized time. we assume that there is no key value of $-\infty$ currently in the Fibonacci heap.

FIB-HEAP-DELETE (x, n)

1. FIB-HEAP-DECREASE-Key (H, n, $-\infty$)

2. FIB-HEAP-EXTRACT-Min (H)

FIB-HEAP-DELETE makes n become the minimum node in the fibonacci heap by giving it a uniquely small key of $-\infty$: The FIB-HEAP-EXTRACT-MIN procedure then removes node 'n' from the Fibonacci heap.

The amortized time of FIB-HEAP-DELETE is the sum of the $O(1)$ amortized time of the FIB-HEAP-DECREASE-key & the $O(D(n))$ Amortized time of FIB-HEAP-EXTRACT-MIN.

Since h

Decrease Key of element x to K.

Case 1:- min heap property not violated

* Decrease Key of x to K

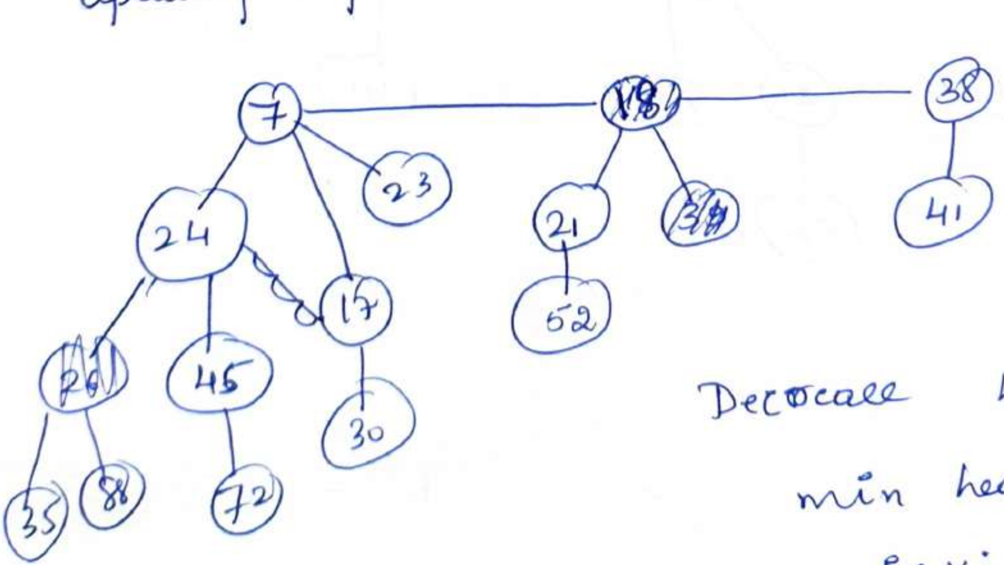
* Change heap min pointer if necessary

Re: decrease key 46 to 45 min heap property is not changed

Case 2:- Parent of x is unmarked

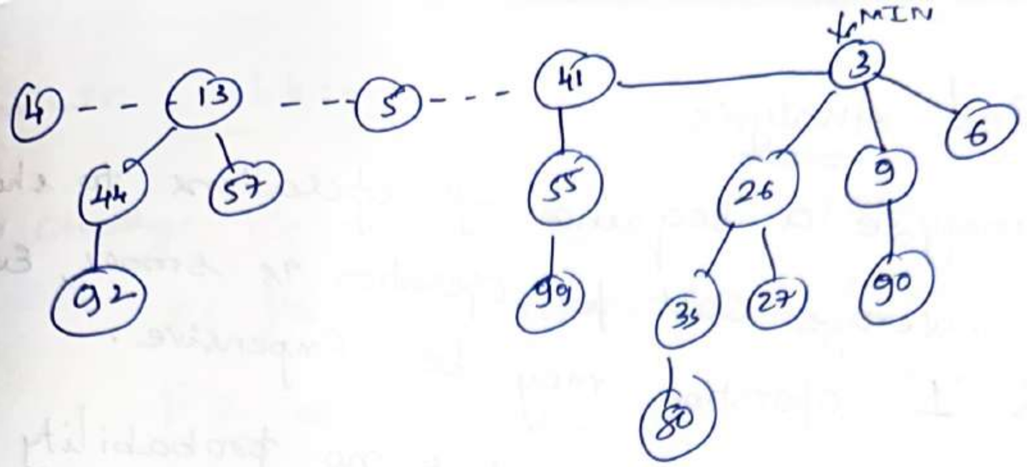
* decrease key of x to K
* Cut off link b/w x & its parent (Min heap property violated)
* mark parent
* add tree rooted at x to root list

updating heap min pointer.



Decrease 45 to 15

min heap property is violated



Delete Node :- (Fibonacci Heaps)

- Delete node x.
  - Decrease key of x to $-\infty$
    - Delete min element in heap

Amortized cost. $O(D(n))$

- $O(1)$ for decrease key
- $O(D(n))$ for delete min
- $D(n)$ = max degree of any node in fibonacci heap

Marked & unmarked node in fibonacci heap

The marking step in the Fibonacci heap allows the data structure to Count how many Children have been lost so far.

An unmarked node has lost no Children, & or a marked node ~~loses~~ ~~another chi~~ has lost one child.

Once a marked node loses another child, it has lost two children & thus needs to be moved back to the root list for reprocessing

EXTRACT_Min: -