RITIKA KUMARI
A20414073

HOMEWORK-3

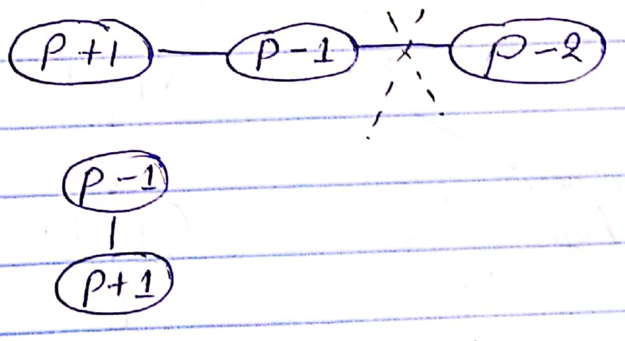
Question 1:

Sol^n: ~~A~~ Let's assume that there is an empty fibonacci heap H. We will create a linear chain of fibonacci heap by taking n nodes which will have the height of (n-1).

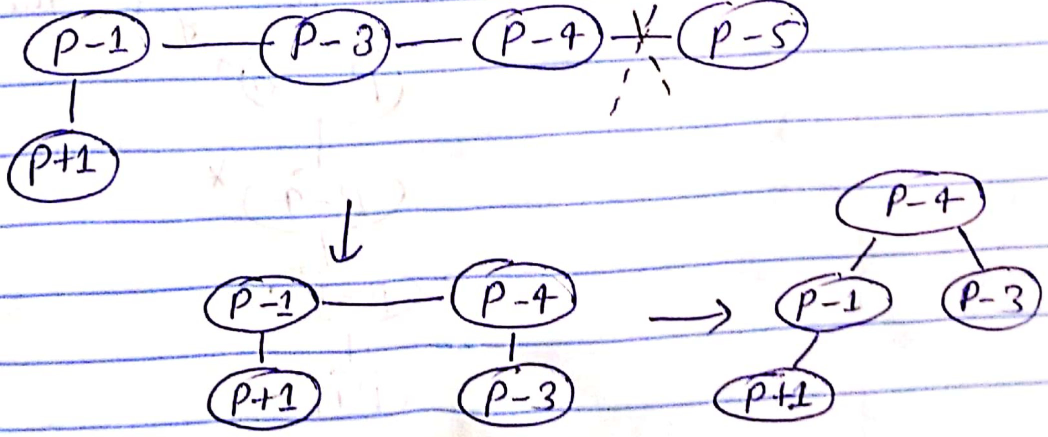
Below are the steps to create fibonacci heap of n nodes:

Step 1: ~~Add~~ Insert three nodes and then delete the least value node which will allow the consolidation of two nodes.

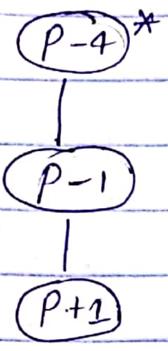
example:



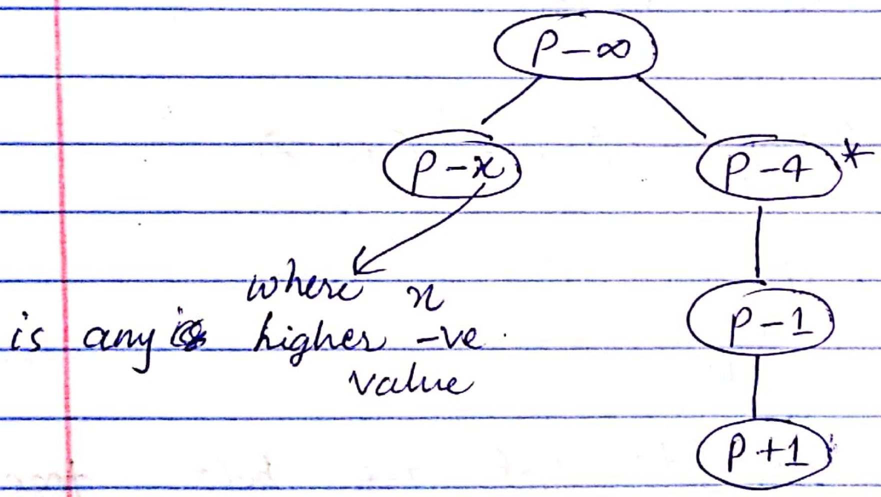
Repeat this to get the below trees



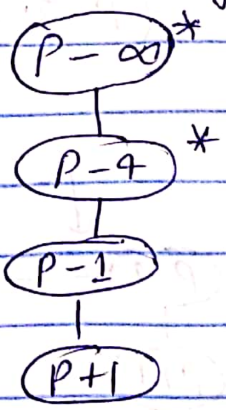
Step 2: Delete (P-3) node and mark the node (P-4).



Step 3: Repeat the step 1 & 2 i.e,
Insert three nodes and deletemin of three nodes to consolidate the tree to get the below :



Step 4: Delete (P-x) to get the ~~tree~~ below :



Step 5: Continue repeating the above steps ie, adding three nodes then deleting single child of a single remaining tree, you will end up with the getting Fibonacci heap of height (n-1) for any n nodes. proved

Problem 2:

Answer: Total no. of elements = n

1, 2, 3, 4, 5, 6, 7, 8 _ _ _ _ n

First, we will perform Make-set operation. We will get below disjoint sets after performing make-set operations on these elements:

$x = \{1, 2, 3\}\ \{4, 5, 6, 7\}\ \{8, 9, 10, 11, 12\}$
$\{13, 14\}, \ldots$

Each sets will be stored in doubly circular linkedlist.

Now, we will perform Union operation on all the ~~elements~~ above sets and will get a single set $x = \{1, 2, 3, 4, 5, \ldots n\}$

Since elements are get stored in doubly linkedlist, it will take $n/2$ pointers to be changed.

$\therefore$ Time taken for Union operations

$= (O(1) + O(1) + O(1) + \ldots\ n \text{ times}) + (O(1) + O(1) + \underset{\text{pointers}}{\ldots} + \tfrac{n}{2} \text{ times})$

$= n(O(1)) + \frac{n}{2}(O(1))$

$= \frac{3n}{2}(O(1))$

$= O(1)$ since n is const.

The time taken for Find-operation will change.

Print 1 element, it will take $O(1)$ time.

$\therefore$ For n elements, time taken for print-set operations $= O(1) + O(1) + O(1) + O(1) + \ldots + n$ times

$= n(O(1))$

$= O(1)$

$\therefore$ ~~The printset~~ for printing n elements it will be always multiple of $O(1)$ and the multiplication factor is always going to be a constant value. So, we can say that the running time for Print-set(x) will be always a linear.

Print-set (x) will not affect any other operation because it is just pointing the elements of set x.

~~PSE~~ <u>PSUEDO-CODE</u>

```
MAKE-SET(X)
{
  → Create Set(x)
  → x can not be defined in another
        set if it is a part of one set.
  → x act as representative of set(x).
}

UNION (x, y)
{
  → Consider two disjoint sets:
        $S_x = x$ , $S_y = y$
  → $S = S_x \cup S_y$
  → Select new representative of set S.
  → Destroy $S_x$ and $S_y$.
}

PRINTSET(x)
{
  → print all elements in Set(x)
{
```

```
FINDSET(x)
{
        return pointer to the representative of set contains x
}
```

Problem 3.

Answer: Total no. of elements in given DataStructure = n

" " " Operations " " = m

Total no. of make-set operations ~~perme~~ performed = n.

∴ Time taken for make-set operation for 1 element = $O(1)$

∴ Time taken for make-set operation for n elements = $n \cdot O(1)$
$= O(1)$

Total no. of union operations performed = $n-1$

Time taken for union-operations for n element = $O(\alpha(n))$

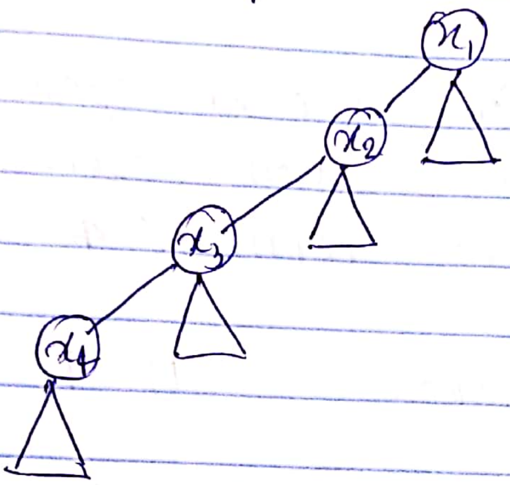
where $\alpha(n) \geq 4$

∴ The value of time complexity can be in between $O(0), O(1), \ldots, O(4)$

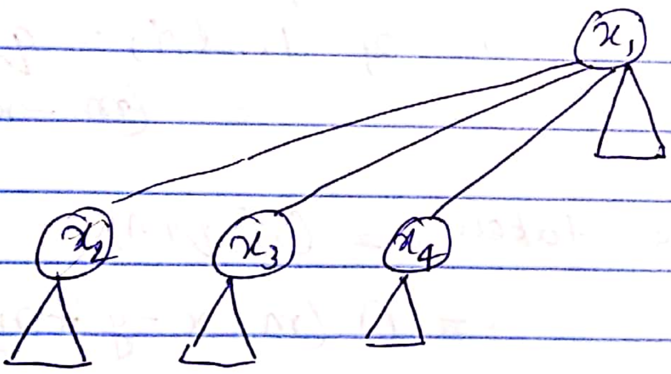
Now, for performing find(i) with path compression, consider the

$= O(q+n)$

below example :



We can easily see that the level of x_1, x_2, x_3 & x_4 will be identical once we perform path-compression.



No. of find(i) operation = q (given)
No. of edges where we perform find operation = n − 1

∴ Total time taken for performing Find(i) operation = O(No. of operations + No. of edges)

$= O(q + (n-1)) = O(q+n-1)$
$= O(q+n)$

Total operations performed = m (given)

Time-taken for Make-set operation = $O(1)$
" " Union operation = $O(\alpha(n))$
where $\alpha(n) \leq 4$ $\forall$ practical purpose

$\therefore$ Total time taken = $O(1) + O(\alpha(n)) + O(q+n)$

= $C_1 + C_2 + O(q+n)$

= $O(q+n)$

Let's assume No. of operation of Make-set = x
" " of Union = y
" " of Find() = q
= $(m - x - y)$

Total time taken = $O(q+n)$
= $O(m - x - y + n)$

Considering x, y and n is constant

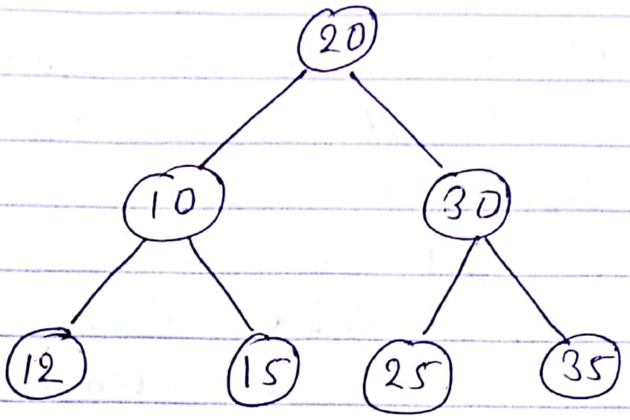
$\therefore$ Total time taken will be $O(m)$

Proved

Scanned by CamScanner

Problem 4:

Answer: Let's consider the given binary tree below.



The above binary tree can be written as-

12	10	15	20	25	30	35

Take the node 20 for the consideration
For successor of a node, we ~~traverse~~ go to the right tree and then traverse to the leftmost node of right tree.
Here, 25 will be the successor of 20 since it is the leftmost element of the right subtree.
If the leftmost element of right tree has the next left child then it will not be the minimum element. Hence we can say that ~~the border having~~ ~~left~~ successor will not have any left child.

For the predecessor, we will go to the ~~lefta~~ left tree of the node and then traverse to the rightmost element.
So, 15 will be the predecessor of 20 since it ~~it~~ is the rightmost element of left subtree.
If the rightmost element has the next right child then it will not be the maximum element.
Hence, we can say that the predecessor of a node can not have any right child. Proved