Abhijuya K·G

16/12/20

IBM18CS004 Binomial Heap

```
// Insertion
void    insert(int x)
{
        // we have to create new node and perform
          union of this node with root
        root = unionHeaps(root, createNode(x));
}

// Delete
void    delete(Node h, int val)
{
        if (h == NULL)      // if heap is empty
            return NULL

        decreaseKeyHeap(h, val, INT_MIN)
        return  extractMin(h);
}

Node extractMin(Node h)
{
        // find min value
        int min = h -> val
        Node curr = h
```

```
while ( curr -> sibling != NULL )
{
    if ((curr -> sibling) -> val < min)
    {
        min = (curr -> sibling) -> val
        min_node_prev = curr
        min_node = curr -> sibling
    }
    curr = curr -> sibling
}

// If a single node is present
   then h = NULL

// else remove min node
else
    min_node_prev -> sibling = min_node -> Sibling

if ( min_node -> child != NULL )
    reverselist ( min_node -> child )
    (min_node -> child) -> Sibling = NULL

// Do union of root h and child
}
```

```
// decrease key
void decreasekey (Node h, int old, int new)
{
    // check de present or not
      and return if not present

    node -> val = new -> val
    Node parent = node -> parent

    while (parent != NULL && node -> val < parent -> val)
    {
      swap (node -> val, parent -> val)
      node = parent
      parent = parent -> parent
    }
}
```