16/12/2020
Wednesday

ADS

Arjun A.S
IBM18CS019

Binomial Heaps

```
// Decrease key by new value in Binomial Heap

Void decreaseKeyBheap(Node * H, int old_val, int new_val)
{
    //1. check whether element is present or not.
    //2. Return if node is not present.
    //3. Reduce value To minimum
    //4. Update heap according to reduced value

    Node * node = findNode (H, old_val);
    if (Node == NULL)
        return;

    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;
    }
}
// function to search for an element
Node * findNode (Node * h, int Val)
{
    if (h == NULL)
        return NULL;
    if (h -> val == Val)
        return h;
    Node * res = findNode (h -> child, val);
```

Arjun A.S
1BM18CS019

```
    if (res != NULL)
        return res;
    return findNode(h->sibling, val);
}

// function to delete an element from Binomial heap
Node * binoDelete(Node *h, int val)
{
    //1. check if heap is empty or not
    //2. Reduce the value of element to minimum
    //3. Delete the minimum element from heap

    if (h == NULL)
    return NULL;

    decreaseKeyBheap(h, val, INT_MIN);
    return extractMinheap(h);
}
// function to extract minimum value from binomial heap
Node * extractMinheap(Node * h)
{
    if (h == NULL)
        return NULL;

    Node * min_node_prev = NULL;
    Node * min_node = h;
    int min = h->val;
    Node * curr = h;
    while ((curr->sibling)->val < Min)
    {
        if (curr->sibling != NULL)
        {
            min = (curr->sibling)->val;
            min_node_prev = curr;
            min_node = curr->sibling;
        }
        curr = curr->sibling;
    }
```

Arjun A.S

Arjun A.S
IBM18CS019

```
if (min_node_prev == NULL && min_node -> sibling == NULL)
        h = NULL;

else if (min_node_prev == NULL)
        h = min_node -> sibling;

else
    min_node_prev -> sibling = min_node -> sibling;

if (min_node -> child != NULL)
{
        revertlist (min_node -> child);
        (min_node -> child) -> sibling = NULL;
}
return unionBheaps (h, root);
}
```