ADS

1BM18CS00-1
Dhanvey K

Binomial Heap

Deletion and decrease Key

```
//Decrease Key by new value.
Void decreaseKeyBheap (Node *H, int old_val, int new_val) {

        Node * Node = find Node (H, old_val);
        if ( Node == NULL)
                return;
        node -> value = 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 delete an element.
Node * Bino_Delete (Node *h, int val) {

        if (h == NULL)
                return NULL;

        decrease_Key Bheap (h, val, INT_min);
        return Extract_min_heap(h);
}
```

//To Extract min_value

```
Node * ExtractMin_Bheap( 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 ->.sibling ; }
        if (min_node_prev = =NULL. && min_node ->sibling
                    ==NULL)
                h=NULL;
        elseif ( min_node_prev ==NULL).
                h= min_node->sibling .

        else.
            min_node_prev -> sibling=min_nodesibling
        if (min_node-> child .!=NULL) {
                revertList (min_node->child)
                (min_node->child)->sibling =NULL;
        } return. unionBheaps.(h,root);
};
```