Khrithik.S.Anand

1BM18CS046

papergrid

Date: / /

ADS. LAB
Binomial Heap Writeup

```
function delete ( Node *h , int val){
        if(!h)
            return NULL;
        decreasekeyBheap (h, val, Int_MIN);
        return extract minbheap(h);
}
function decreasekey BHeap (Node *H , int oldv , int newv){
            Node * node Find (H, oldv);
            if (!Node ) return;
            node->val =newv;
            Node * parent = node -> parent;
            while( parent) =NULL && nod -> val <parent){
                Swap (node -> val, parent ->val)
                Node=parent;
                parent =parent -> parent;
            }
}

function *extract minHeap (Node *h){
            if(!h) return NULL;
            Node *min_pos =NULL;
            Node *min -h=h
            int min =h->val;
            Node *curr =h;
            while (curr->sibling ! =NULL){
                if ((curr-> sibling) -> val<min){
                    min = curr -> sibling->val;
                    min_prev =curr;
                    min = curr->sibling;
                }
```

Khrithik S. Anand

1BM18CS040

papergrid

Date: / /

```
        curr = curr -> sibling;
    }
    if (min-prev = NULL && min -> sibling == NULL) h = NULL;
    else if (min-prev == NULL) h = min -> sibling;
    else min-prev -> sibling = min sibling;

    if (min -> child) {
        revertlist (min -> child);
        min -> child -> sibling = NULL;
    }
    return unionBheap (h, root);
}
function findNode (Node *h, int val) {
        if (!h) return NULL
        if (h -> val == val) return h;
        Node * res = findNode (h -> child, val);
        if (res != NULL) return res;
        return findNode (h -> sibling, val);
}
function revertlist (Node * h) {
        if (h -> sibling) {
            revertlist (h -> sibling);
            h -> sibling -> sibling = h;
        } else root = h
}
```