ADS LAB

AKSHAY MITTUR

1BM18CS010

09/12/20

BINOMIAL HEAP

Insert Function: (input: head, key)

```
{
=>          Node *temp=newNode(key);
        ~~Ate~~ list<Node*> t;
            t.push_back(temp);
            t =unionBH(_head, t);
            return adjust(t);
}

adjust( list<Node *> heap){
        if (heap.size ~~=~~ <= 1) return heap;
        list< Node * > new_heap;
        auto it1, it2, it3;
        it1 = it2 = it3 = heap.begin();
        if (heap.size()==2) {
                it2=it1;
                it2++;
                it3 = heap.end();
        } else {
                it2++;
                it3=it2;
                it3++;
        } while(it1 != heap.end()) {
                if (it2 ==heap.end()) it1 ++;
            else if( *it1->degree < *it2->degree) {
                    it1++, it2++;
                    if (it3!= heap.end()) it3++;
                }
```

M.. Akshay Mirta
1BM18CS010
ADS LAB

9/12/20

```
        else if (*it1->degree == *it2->degree){
                Node *temp;
                *it1= merge(*it1, *t2);
                it2= heap.erase(it2);
                if(it3!= heap.end()) it3++;
        }
        else if(it3!=heap.end() && *it1->degree== *it2->degree &&
                    *it->degree == *it3->degree) {
                    *it++, it2++, it3++;
        }
    return heap;
}

Function GetMin (list<Node *> heap){
            auto it= heap.begin();
            while (it!= heap.end()) {
                    if (*it->data < temp->data) temp= *it;
                it++;
            }
        return temp;
}
Function extract-min( list<Node*heap){
    list <Node*> new_heap, lo, Node *temp;
        temp=getMin(heap), auto it =heap.begin();
        while (it != heap.end() {
                if (*it != temp) new_heap.push_back(*it);
            it++;
        }
    lo = rem(temp);
    new_heap = unionBH(new_heap, lo);
    new_heap = adjust(new_heap);
    return new_heap;
}
```