ADS Lab

```
func adjust (Node head, key):
    if head.size <= 1 return heap
    Node new_heap
    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 != head.end: it3++;
        else if (it3 != head.end() && (*it1)->degree
            == (it2)->degree && (*it3->degree)):
            it1++; it2++; it3++;
        else if (it1->degree == it2->degree):
            Node temp;
            it1 = meggeBinomialTree (it1, it3)
            it2 = heap.erase (it2);
            if (it3 != heap.end()) it2++;
    }
}
return heap;
}
```

```
func getMin (Node & heap):
    Node iterator it = heap.begin()
    Node temp : it
    while (it != heap.end()) {
        if (it.data < temp.data):
            temp = it;
    return temp

func extractMin (Node heap):
    Node new_heap, lo;
    Node temp;

    temp = getMin (heap)
    Node iterator it;
    it = heap.~~begin~~ begin()
    while (it != heap.end()):
        if (it != temp) new_heap.push_back (it);
        it++
    lo = removeMin (temp);
    new_heap = union (new_heap.lo);
    new_heap.adjust (new_heap)
    return new_heap;
```