2014-19498 정은주

독어교육과 알고리즘 HW2 merge sort min Heap 구현보고서

```
static PriorityQueue<Node> minHeap = new PriorityQueue<>();
class Solution1 {
    public static class Node implements Comparable<Node> {
        int src;
        int des;
        int val;

        public Node() {
            this.src = 0;
            this.des = 0;
            this.val = 0;
        }
        public Node(int src, int des, int val) {
            this.src = src;
            this.des = des;
            this.val = val;
        }
        @Override /** priority queue의 Comparable 이용, min Heap구현 **/
        public int compareTo(Node another) {
            if(this.val < another.val) return -1;
            else return 1;
        }
    }
```

JAVA 내장 library인 PriorityQueue를 사용해 minHeap을 구현한다. 만든 minHeap은 우선, 16개의 Node를 담게 되는데, 이 Node 내부에는 각각 자신의 위치를 알 수 있는 src index와 해당 구간이 끝나는 des index를 각각 가지며, 출발하는 위치의 value 값을 가지게 된다. 따라서 담기는 16개의 Node는 array A의 값을 복사한 array copied 내부에서 16등분 된 부분 array 16개를 각각 대표하고, 그 내부에서만 움직이는 값이 된다. 이때 comparable<Node>를 implement해서 minHeap을 구현한다. compareTo 함수를 Override하여서 Node의 value 값들을 비교해 작은 값이 루트에 오도록 즉, 가장 작은 값이 poll() 될 수 있도록 하였다.

```
        for(int i = src; i<src+15*q; i=i+q) {
            minHeap.offer(new Node(i, i+q-1, copied[i]));
        }
        minHeap.offer(new Node(src+15*q, des, copied[src+15*q]));
        /** Merge: Theta(n) time complexity **/
        for(int i = src; i<=des; i++) {
            nw = minHeap.poll();
            temp[i] = nw.val;
            /** choosing minval -> constant time using Heap **/
            minHeap.offer((nw.src< nw.des)?
                new Node(nw.src+1, nw.des, copied[nw.src+1]):
                new Node(nw.des, nw.des, Integer.MAX_VALUE));
        }
```

1)16개의 부분 array들의 가장 작은 값들을 나타내는 node를 src, des, val을 매달아 min Heap priorityQueue에 넣는다. 2)총 n번(des-src)만큼 가장 작은 값을 minHeap에서 poll() 하고, 그 값의 value를 정렬하려는 array에 (temp[]) 집어넣는다. 그 값이 src index가 des index보다 작다면, 즉, 부분 array의 범위를 벗어나지 않는다면, 해당 노드의 src index++하여 new src, des, val을 적절히 매달아 다시 Heap에 집어넣고, array 범위를 벗어났다면, MAX_VALUE를 매달아 집어넣는다. 이렇게 총 n번의 minHeap.offer() 과 minHeap.poll() 과정을 반복하면 temp[] array에는 값들이 increasing order로 sort되어 정렬된다.