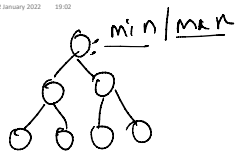
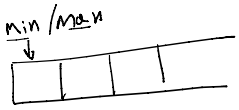
Priority Queue using Heap

12 January 2022 19:02





min/max- Heap

min/max Priority Queue

[10, 20, 30, 40, 50, 60, 35,]

0 1 2 3 4 5 6

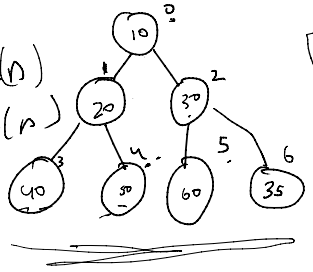
PQ =)

Peek() =) O(1)

add() =) log(n)

remove() =) log(n)

Size() =) O(1)



$l_i = 2 \times i + 1$

$r_i = 2 \times i + 2$

$p_i = \frac{(i-1)}{2}$

heapify ↓ down / up

⇒ Tree should be CBT till (h-1)

⇒ Parent should always be smaller than both children

PQ {

AL<Int> data ;

upheapify(i) {

pi = (i-1)/2 ;

if(i ≥ 0 && data[i] < data[pi]){

swap (i, pi) ;

upheapify (pi) ;

}

}add(d) {

data.add(d);

upheapify();

}

downheapify (i) {

res = i ;

li = 2*i + 1

if(li < data.size && data[res] > data[li])

res = li

ri = 2*i + 2

if (ri < data.size &&

data[res] > data [ri]) {

res = ri

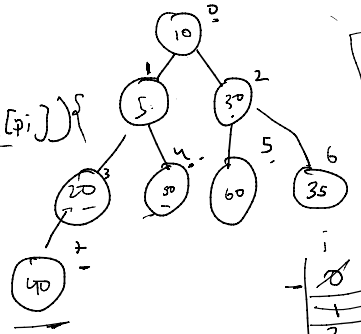
if(res == i) return ;

Swap (res, i)

if (res < data.size downheapify(res);

[10, 20, 30, 40, 50, 60, 35, 5]

0 1 2 3 4 5 6 7



$l_i = 2 \times i + 1$

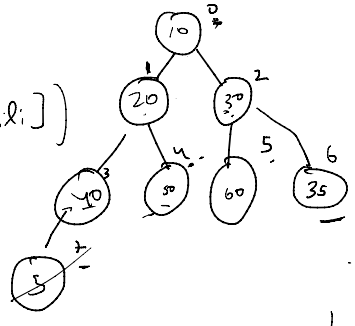
$r_i = 2 \times i + 2$

$p_i = \frac{(i-1)}{2}$

i	p_i
~~0~~	
1	0
3	1
7	3

[40, 10, 30, 20, 50, 60, 35,]

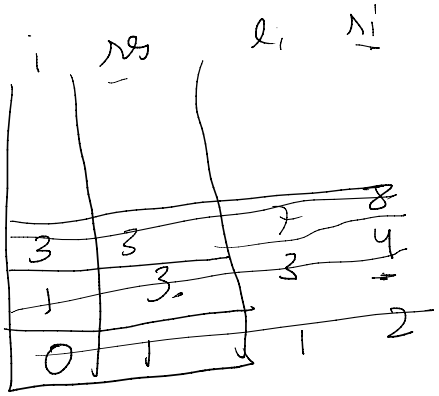
0 1 2 3 4 5 6



$l_i = 2 \times i + 1$

$r_i = 2 \times i + 2$

$p_i = \frac{(i-1)}{2}$



}

remove() {
res = dat[0];
swap(0, size-1);
downHeapify
return res;
}

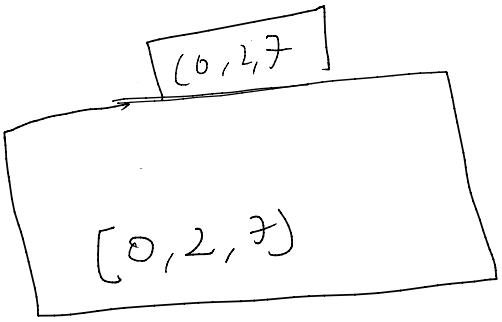
k - Sorted lists

([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

0 1 2

[1, 2, 3, 4, 5, 6

Data {
idx;
lidx;
value;
}



1 - 4 = -3

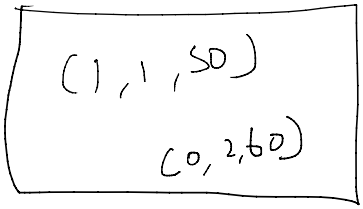
4 - 1 = 3

[[20,30], [40,50], [60,70]]

0 1 0 1 0 1

0 1 2

(0,0,4) - (0,1,1)



d: (0, 1, 40)

idx lidx val

```java
public static ArrayList<Integer> mergeKSortedLists(int[][] arr) {
    ArrayList<Integer> res = new ArrayList<>();
    java.util.PriorityQueue<DataIndex> pq = new java.util.PriorityQueue<>();
    for (int i = 0; i < arr.length; i++)
        pq.add(new DataIndex( idx: 0, i, arr[i][0]));

    while (!pq.isEmpty()) {
        DataIndex d = pq.remove();
        res.add(d.val);
        if (d.idx + 1 < arr[d.lidx].length)
            pq.add(new DataIndex( idx: d.idx + 1, d.lidx, arr[d.lidx][d.idx+1]));
    }
    return res;
}
```

res: [20, 30, 40