Heap [min pq]

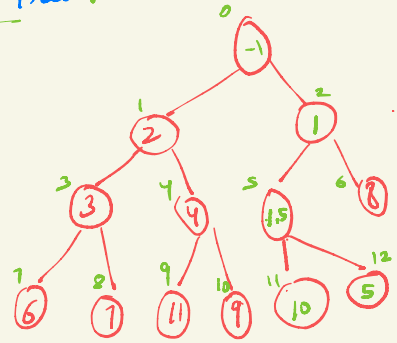
Complete Binary Tree (CBT)

Heap order property

why?
what?
how?

-1

upHeapify → every parent will be smaller than child



{-1, 2, 1, 3, 4, 1.5, 8, 6, 7, 11, 9, 10, 5}

pi = 4

lci = 2*pi+1;

rci = 2*pi+2;

lci = 9

rci = 10

pi = 12-1/2 ⇒ 5

pi = 5-1/2 ⇒ 2

pi = 2-1/2 ⇒ 0

swap(ci, pi)

ci

{pi = ci-1/2}

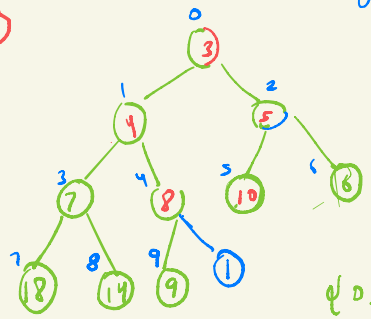
12

```
static class Heap {
    ArrayList<Integer> data;

    public Heap(){
        data = new ArrayList<>();
    }

    public void swap(int i, int j){
        int valAtI = data.get(i);
        int valAtJ = data.get(j);

        data.set(i,valAtJ);
        data.set(j,valAtI);
    }

    public void upHeapify(int ci){
        int pi = (ci-1)/2;

        if(data.get(pi) > data.get(ci)){
            swap(ci,pi);
            upHeapify(pi);
        }
    }
    // add
    public void add(int val){
        data.add(val);

        upHeapify(data.size()-1);
    }
```

data ⇒ {3, 4, 5, 7, 8, 10, 6, 18, 14, 9}

O(n)

-1



down Heapify

{0, 1} pi, minCi

pi = 0

lci = 2*pi+1 = 1

rci = 2*pi+2 = 2

pi = 1

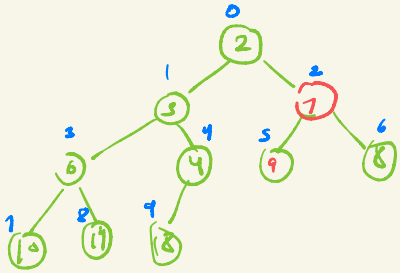
lci = 3

rci = 4

{1, 4}

```java
private void downHeapify(int pi){
    int lci = 2*pi + 1;
    int rci = 2*pi + 2;

    int mini = pi;

    if(lci < data.size() &&  data.get(lci) < data.get(mini)){
        mini = lci;
    }

    if(rci < data.size() && data.get(rci) < data.get(mini)){
        mini = rci;
    }

    if(mini != pi){
        swap(pi,mini);
        downHeapify(mini);
    }
}
// remove
public int remove(){
    if(data.size()==0){
        System.out.println(x: "There is no element in heap");
        return -1;
    }

    swap(i: 0,data.size()-1);
    int rv = data.remove(data.size()-1);

    downHeapify(pi: 0);

    return rv;
}
```

{2, 3, 7, 6, 4, 9, 8, 10, 19, 18}



break till 10:30

mini = 0 1 3 8

pi = 0 1 3

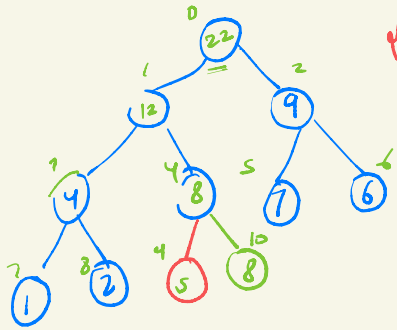
lci = 1 3 7

rci = 2 4 8

{22, 12, 9, 4, 8, 7, 6, 1, 2, 5, 8, 3}

ci = 10 1 0

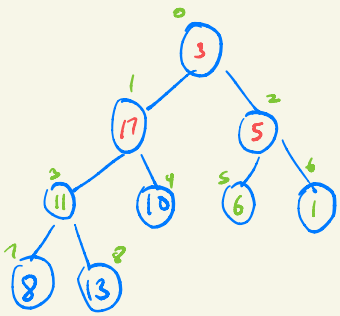
pi = 1 0 -1



```java
private void swap(int i, int j){
    int valAtI = data.get(i);
    int valAtJ = data.get(j);

    data.set(i,valAtJ);
    data.set(j,valAtI);
}

private void upHeapify(int ci){
    int pi = (ci-1)/2;

    if(pi>=0 && data.get(pi) < data.get(ci)){
        swap(ci,pi);
        upHeapify(pi);
    }
}
// add
public void add(int val){
    data.add(val);

    upHeapify(data.size()-1);
}
```

6, 4, 1, 3, 10, 9, 17, 8, 13

HoP



arr => 0 1 2 3 4 5 6 7 8
{3, 17, 5, 11, 10, 6, 1, 8, 13}

PQ

```
void swap(int[] data, int i, int j){
    int temp = data[i];

    data[i] = data[j];
    data[j] = temp;
}

void upHeapify(int[] data, int ci){
    int pi = (ci-1)/2;

    if(pi>=0 && data[pi] > data[ci]){
        swap(data,pi,ci);
        upHeapify(data,pi);
    }
}

void buildHeap(int arr[]) {
    for(int i=0; i<arr.length; i++){
        upHeapify(arr,i);
    }
}
```