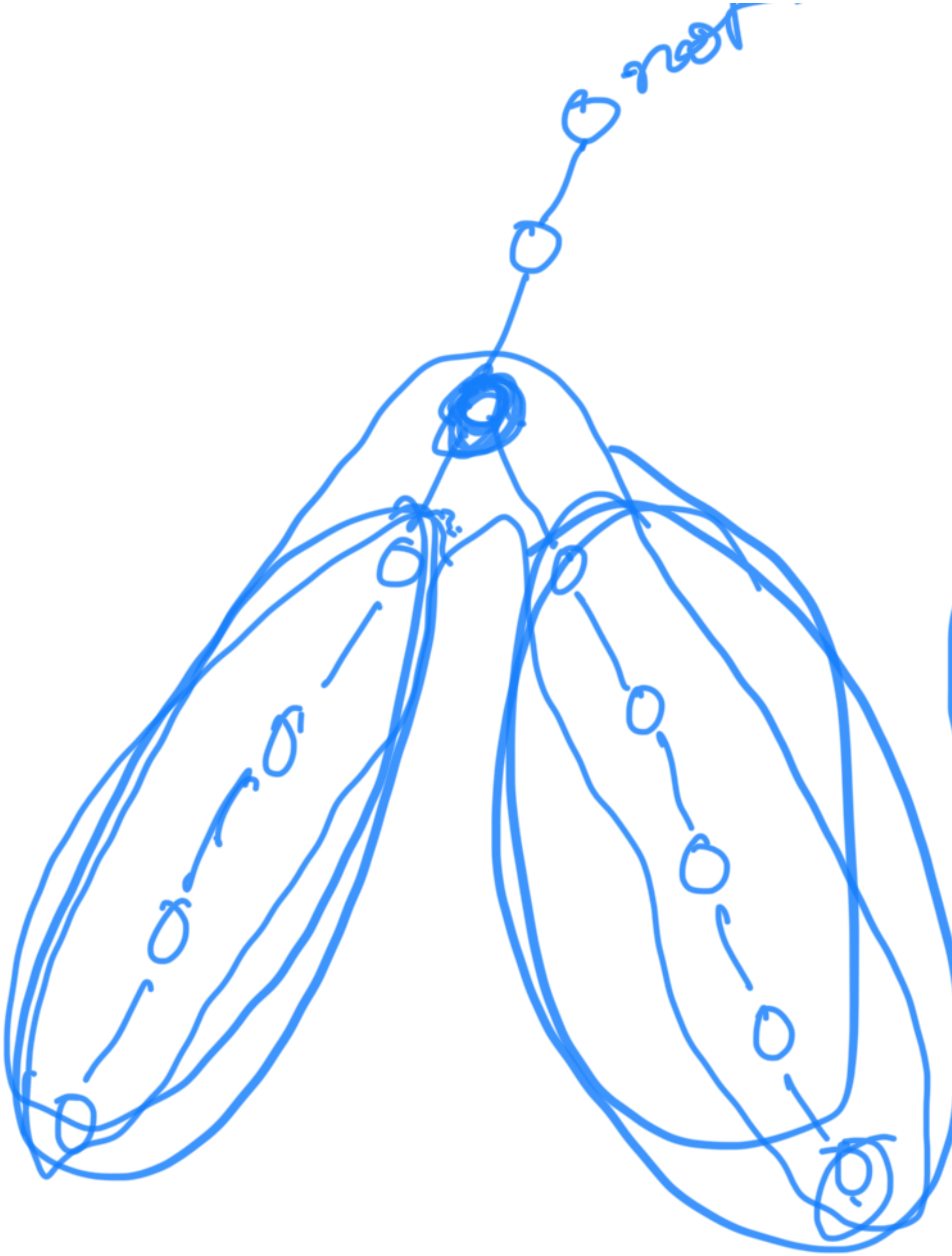
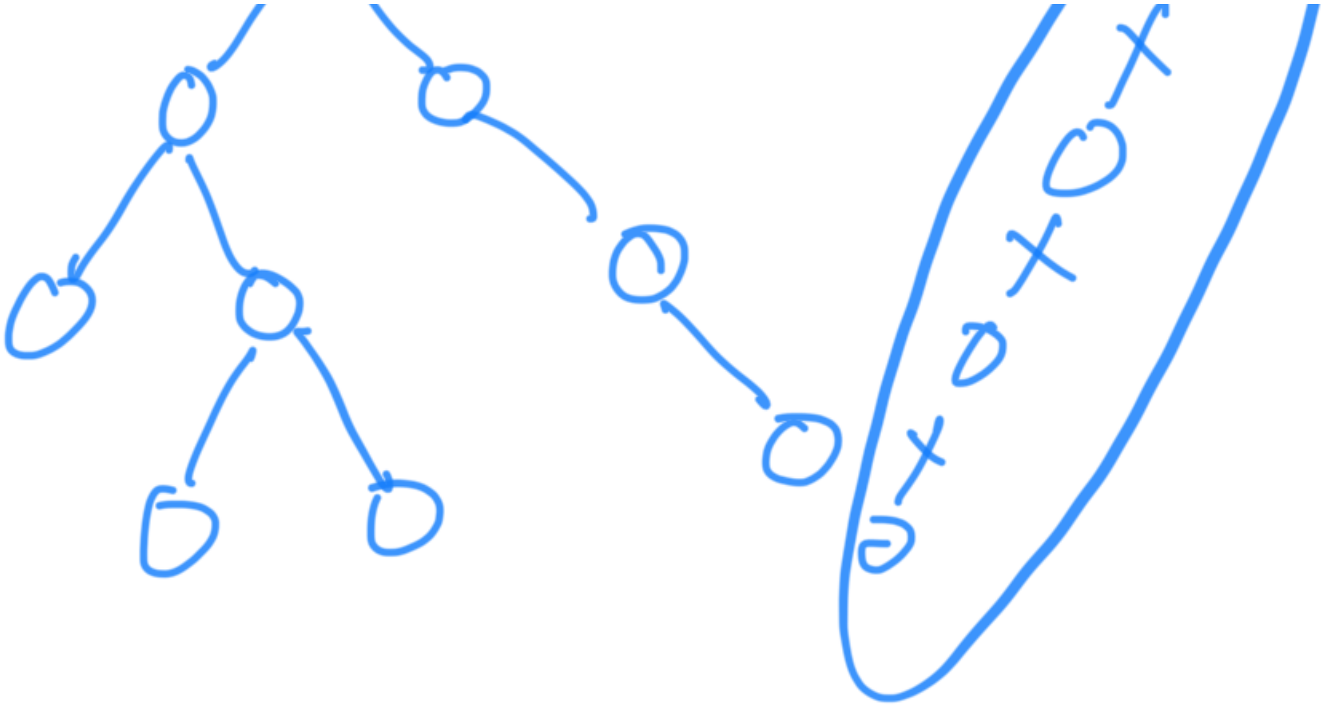


$n * n'$

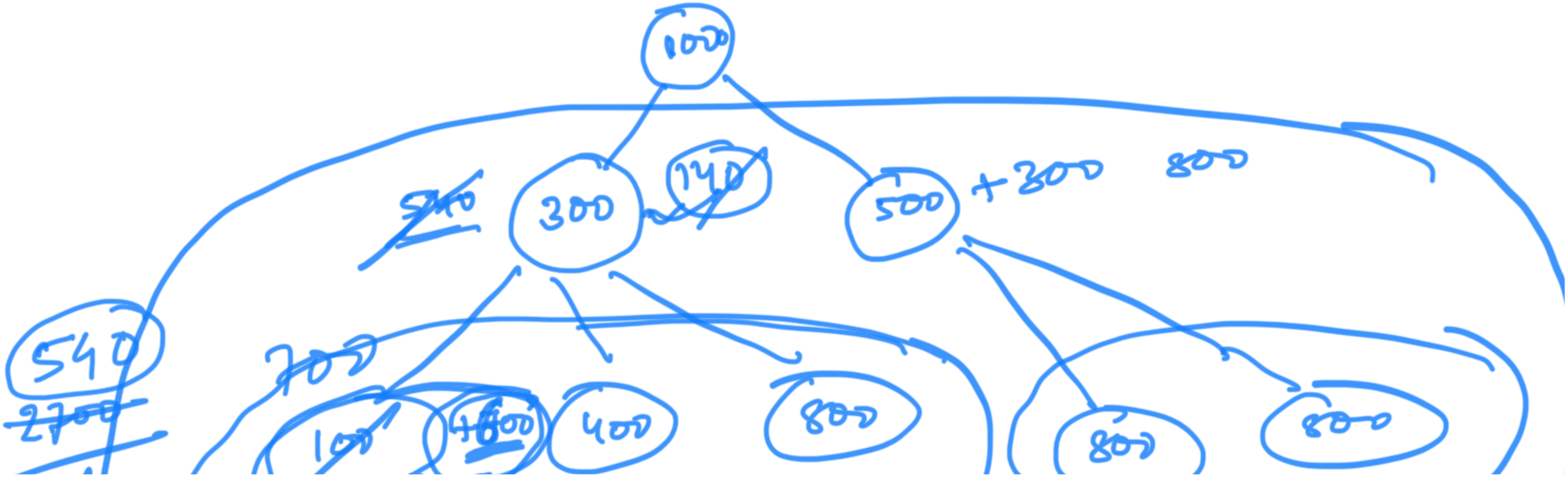
$\boxed{n * n}$

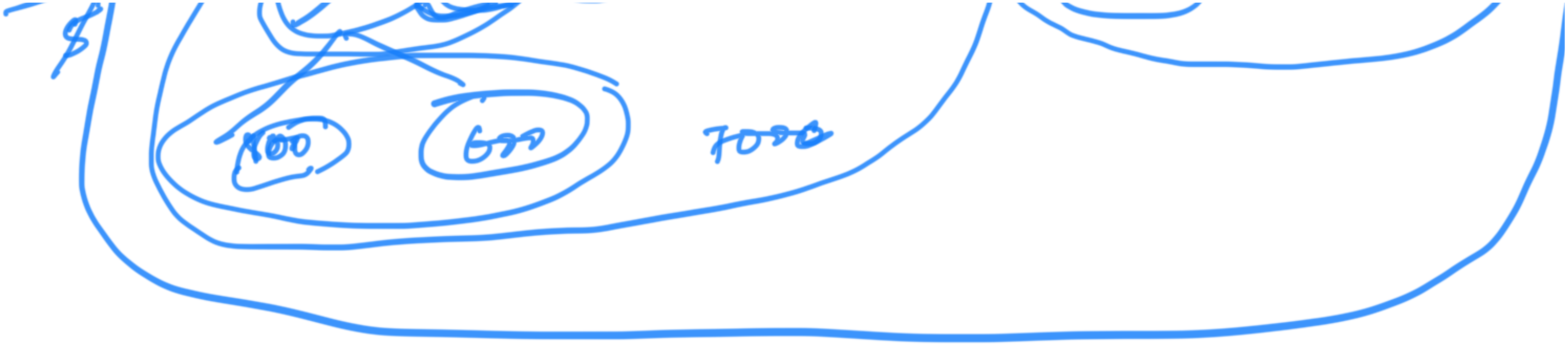
n^2



Given a company hierarchy.

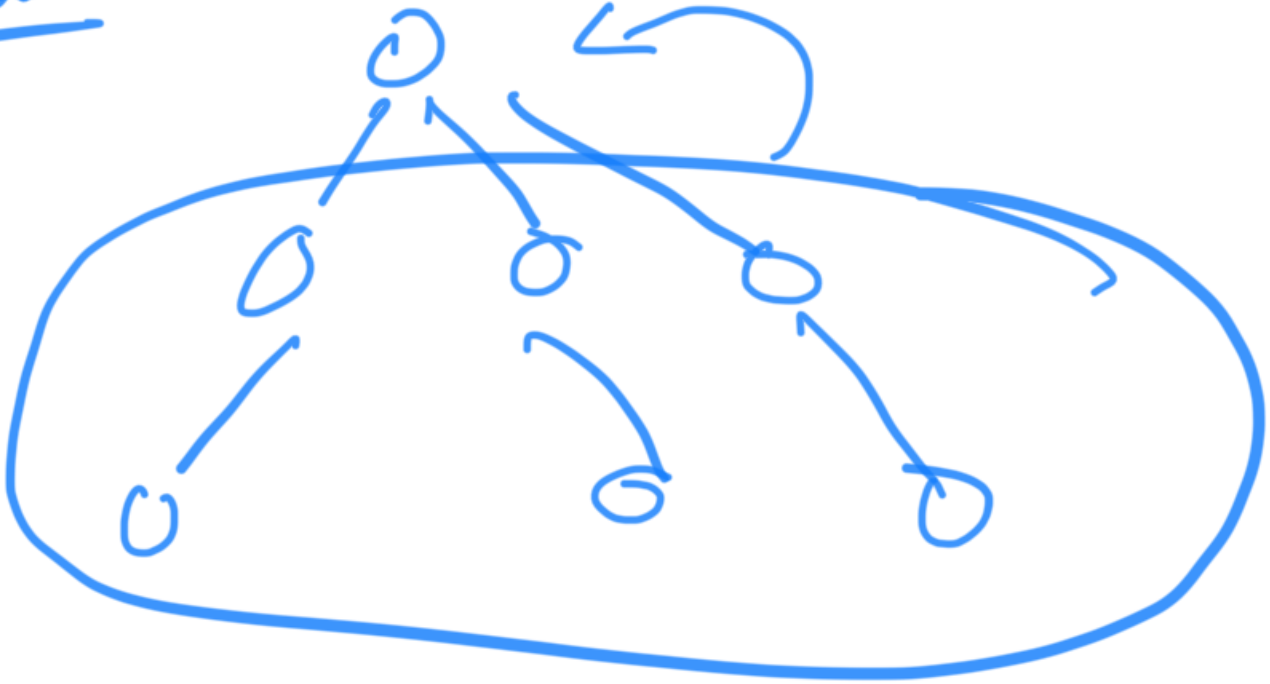
Run an appraisal cycle.





540 + 800 +

Maximum



Avg ✓

max ✓

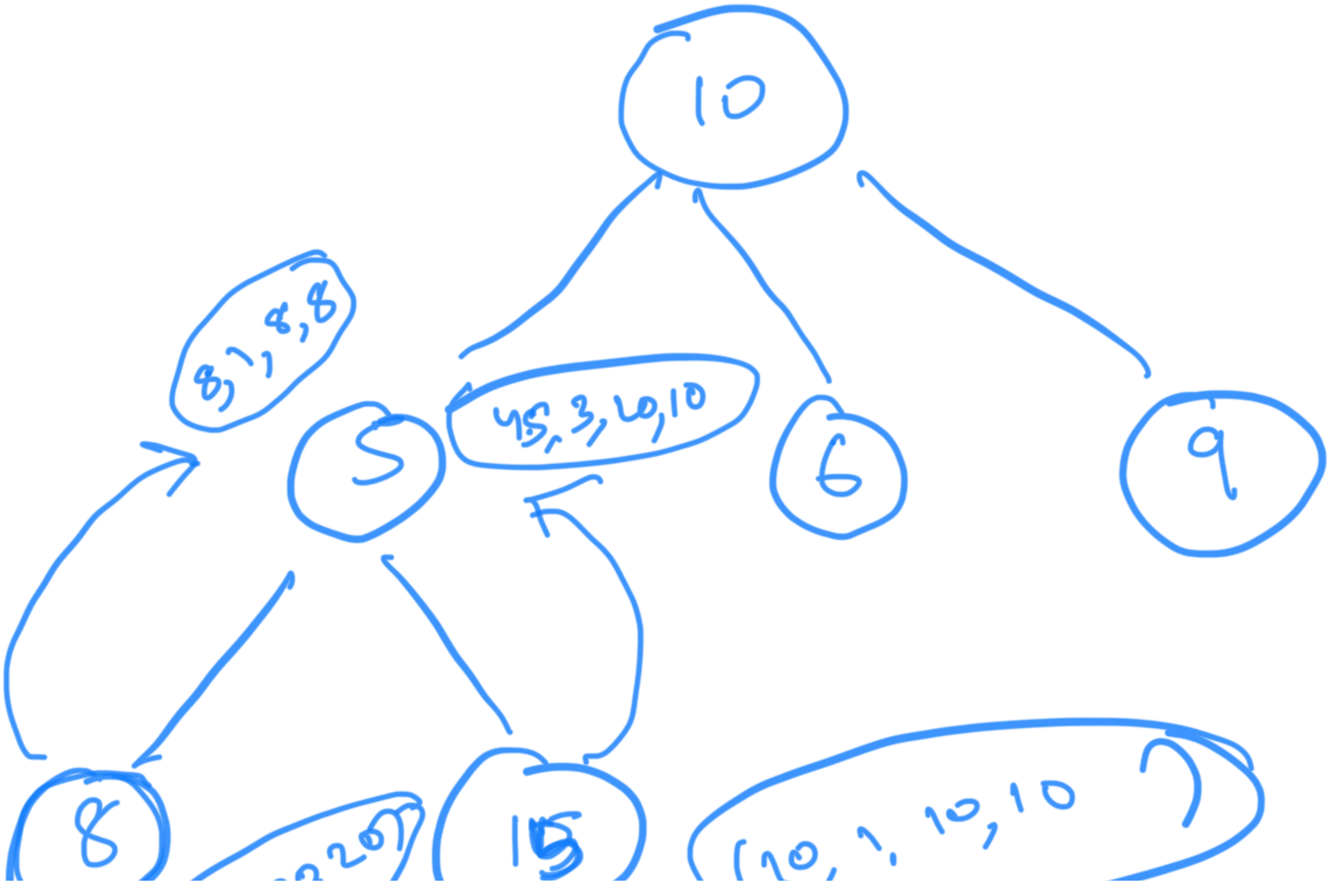
min

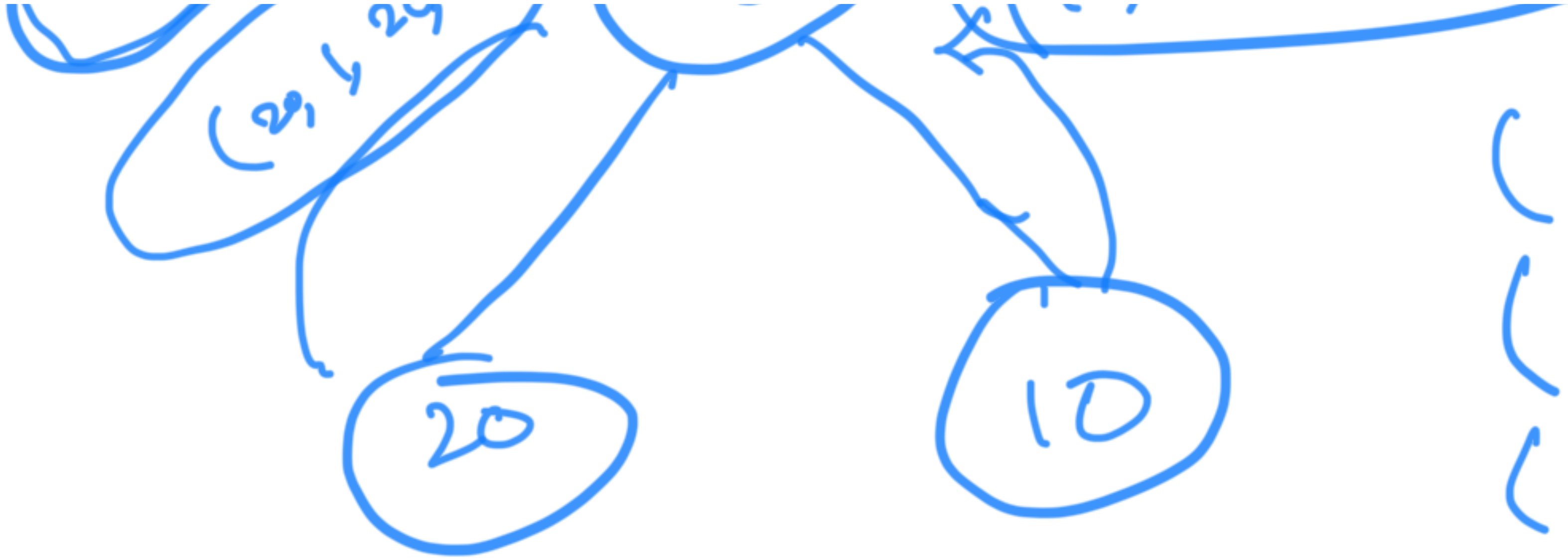
class

```
record stat(int sum, int count, int max, int min);
     int media, int SecMax, int varia
```

```
class TreeNode {
        int salary;
        List<TreeNode> children;

}
```

```
class Appraisal {

     void applyAppraisal(TreeNode root){
}

Stat appraisalHelper ( TreeNode root ){
  if (root = null)
   {    new Stat (0, 0, -Integer.MIN_VALUE,
                            Integer.MAX_VALUE )
                                                       -INF, +INF ):
   }
  Stat ret = new Stat (0, 0, -INF, +INF ):
  for ( TreeNode child : root.children )
  {
     Stat childStat = appraisalHelper (child,

     ret.sum += childStat.sum;
     ret.count += childStat.count;
                          Math.max ( ret.max,
    ret.max = Math.max(ret.max,
                       childStat.max)

    ret.min = Math.min(ret.min,
                       childStat.min;
}
        new Slary = getNewSalary(~~stat~~ ret)
          if(newSlan > root.sala)) }
                                                  }
~~root.salary =~~

int avg = ret.sum / ret.count ;

if( avg > root.salary )
  {
     root.salary = avg;
  }

ret.sum += root.salary;

ret.count += 1 ;

ret.max = Math.max(ret.max,
                   root.salary)
        ret.min = Math.min(ret.min,
                            root.salary)

        return ret;
    }
```





(30, 2, 20, 10)

(~~45~~,

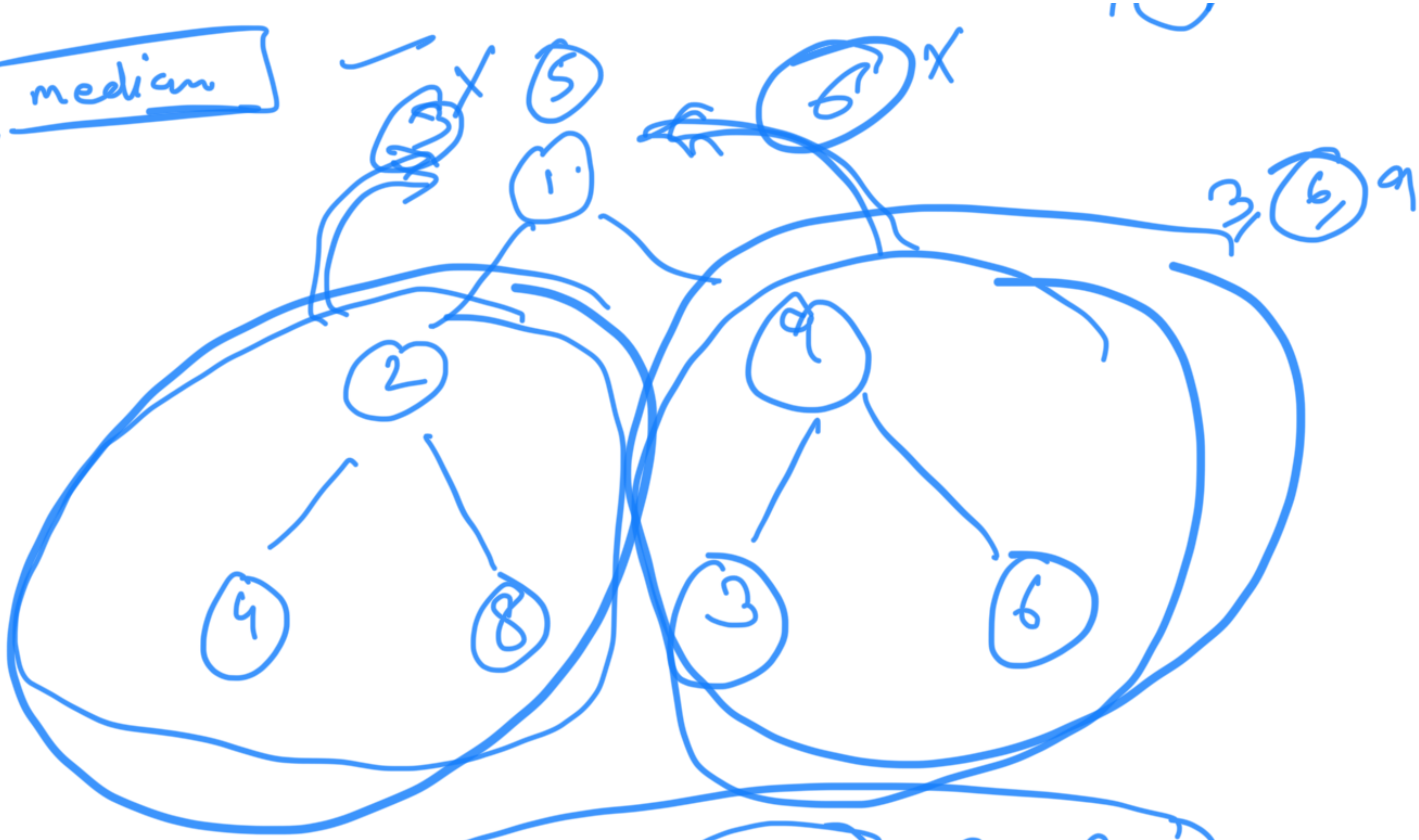
(45, 3, 20, 10)

```
applyAppraisal ( root, ( stat ) → {
                 return stat.sum /
                        stat.count;

            }
      }
```

lambda are light weight strategy

2, 4, 5

median



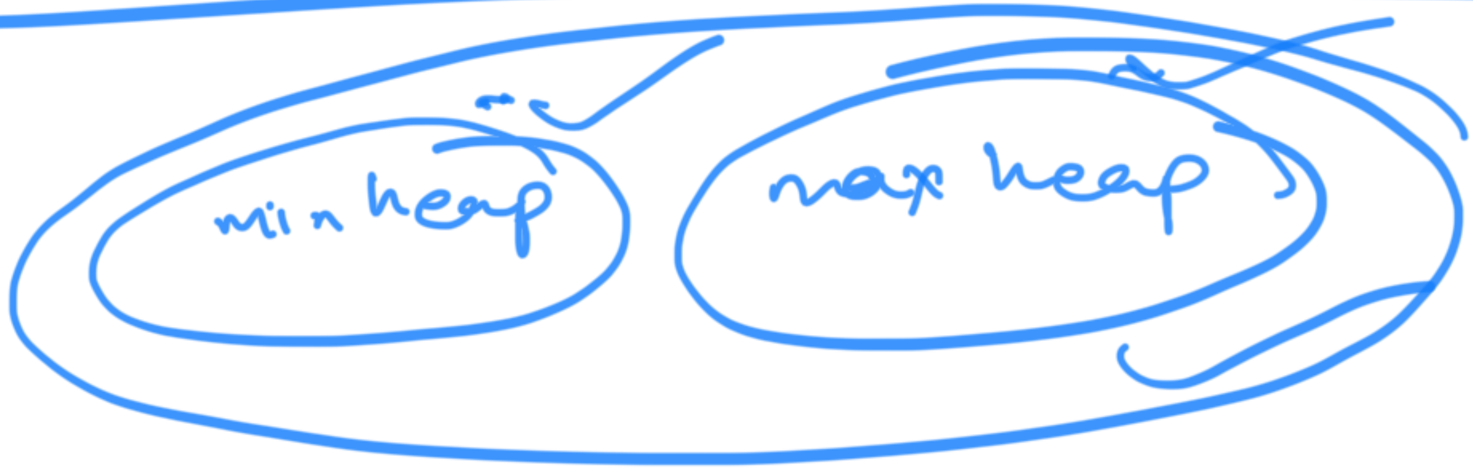
2, 3, 4, 6, 8, 9

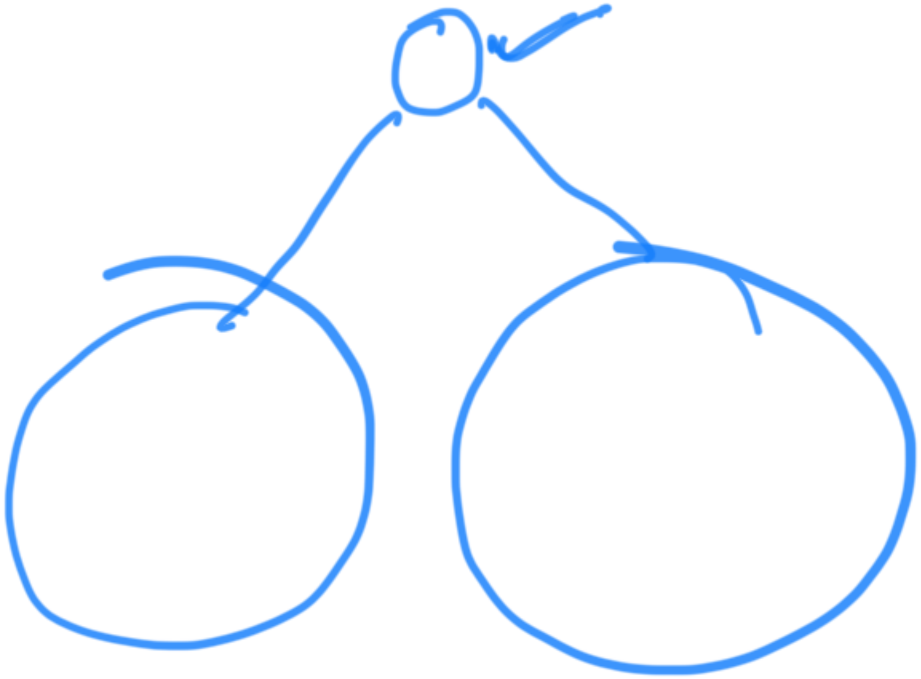
5

O(N)

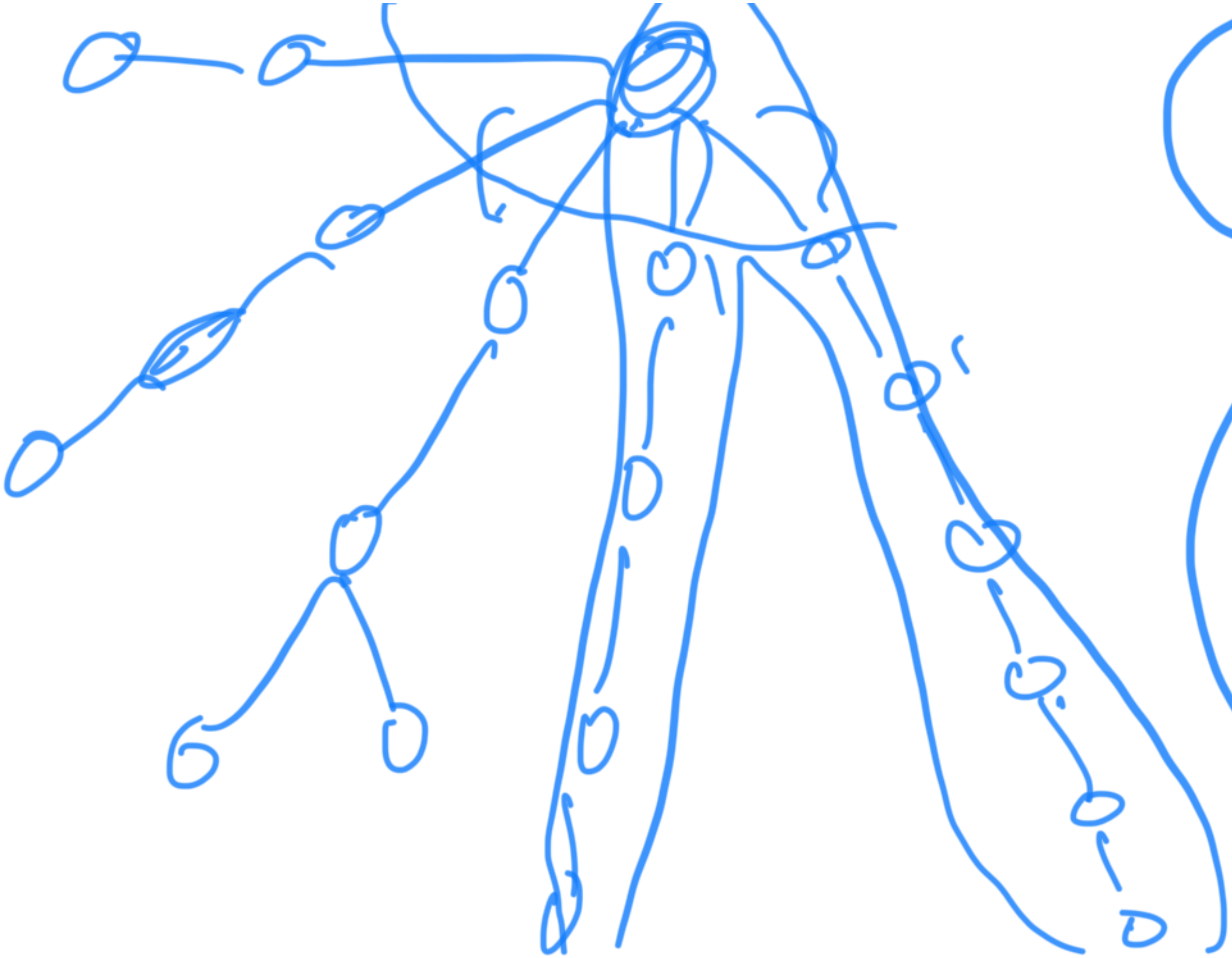
O(N × N × logN)

Median in a stream of no's





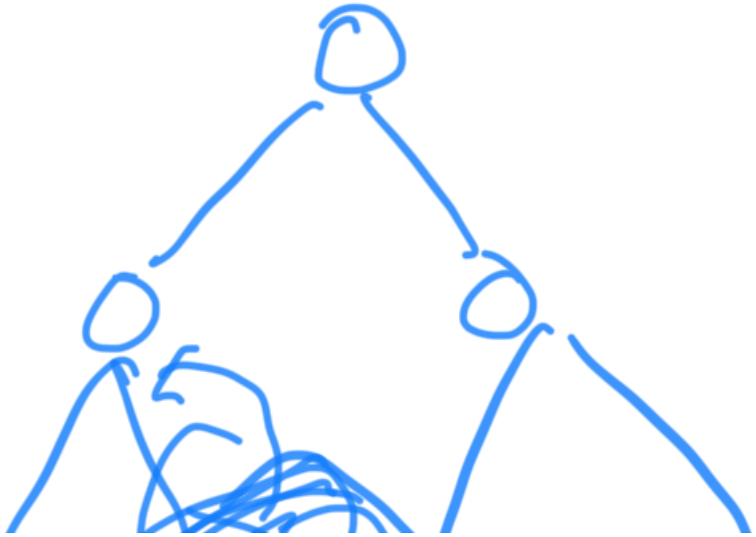
n logn

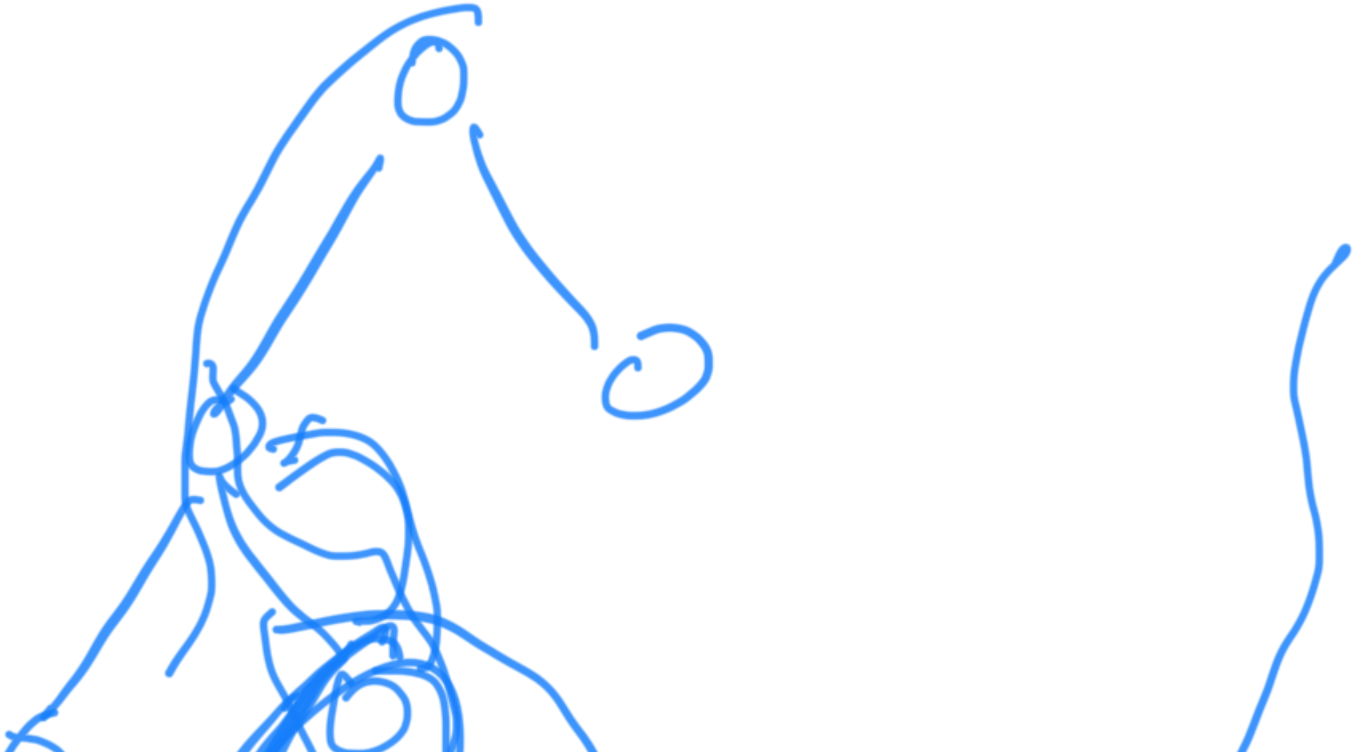


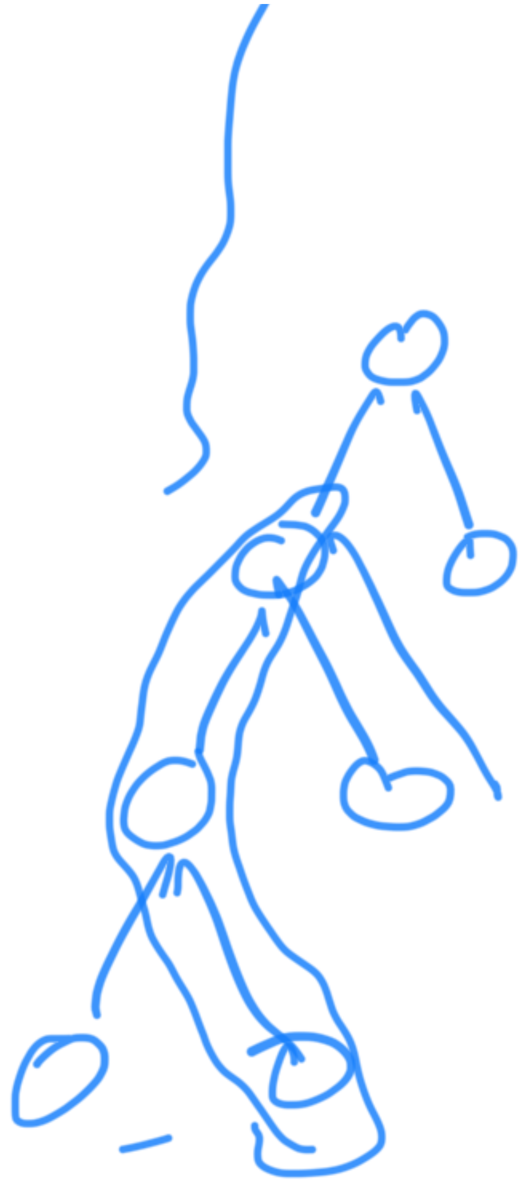
max

seeMax

```
h
h > max
    seeMax = max
    max = h
else
    h > seeMax
        seeMax
```







PathSum

```
LPS = pathSum(root.left)
RPS = pathSum(( root.right);
LPS = LPS < 0 ? 0 : LPS
RPS = RPS < 0 ? 0 : RPS;
max = Math.max( max, root.val + LPS + RPS)
                max
```

```
return root.val + (LPS + RPS)
```

BST validation

post order traversal

```
   max   min
Pair<Int,Int>  isValidBST ( TreeNode root, min
{
      if( root == null )
         {                      }

      if (isleft (root))
       {
          return ( root.val , root.val );
       (
```

)

lr

Return ← isValidBST(root.left) ✓

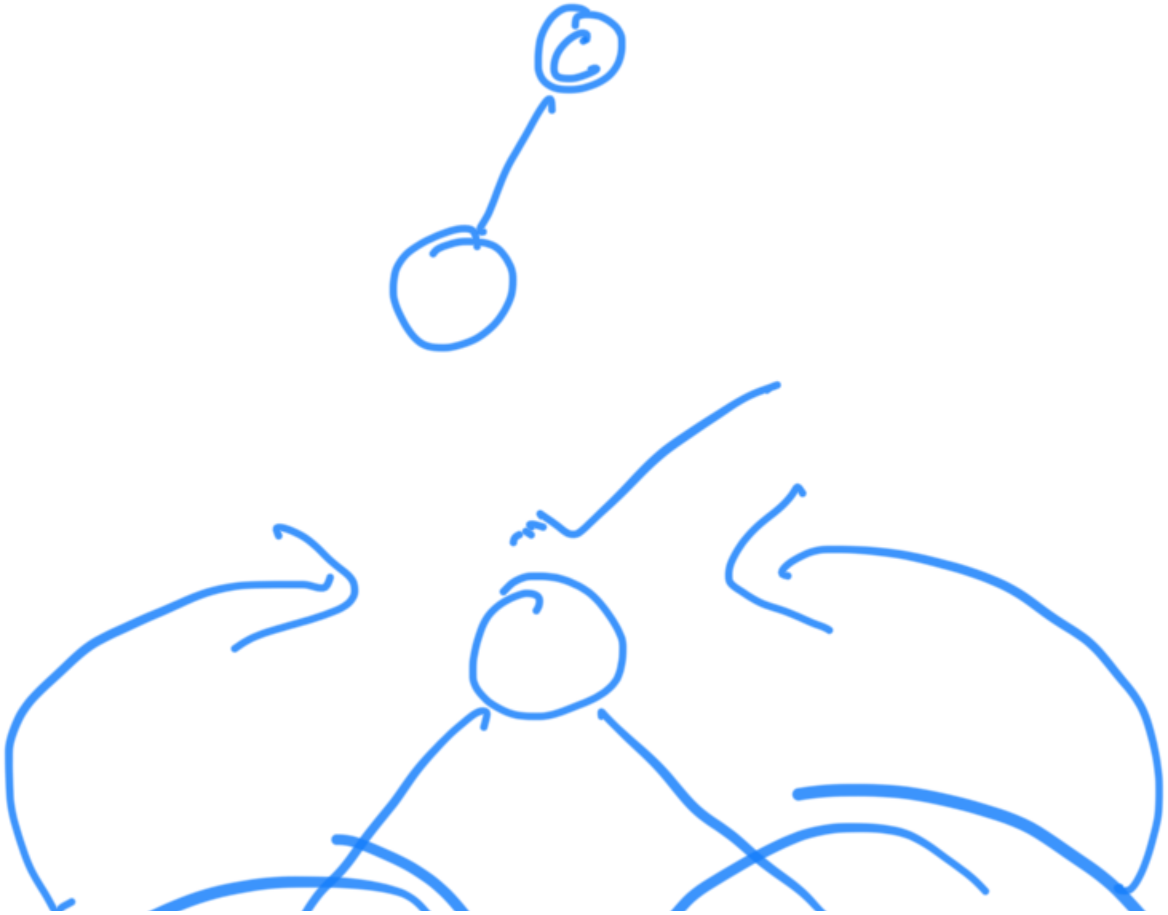
&&

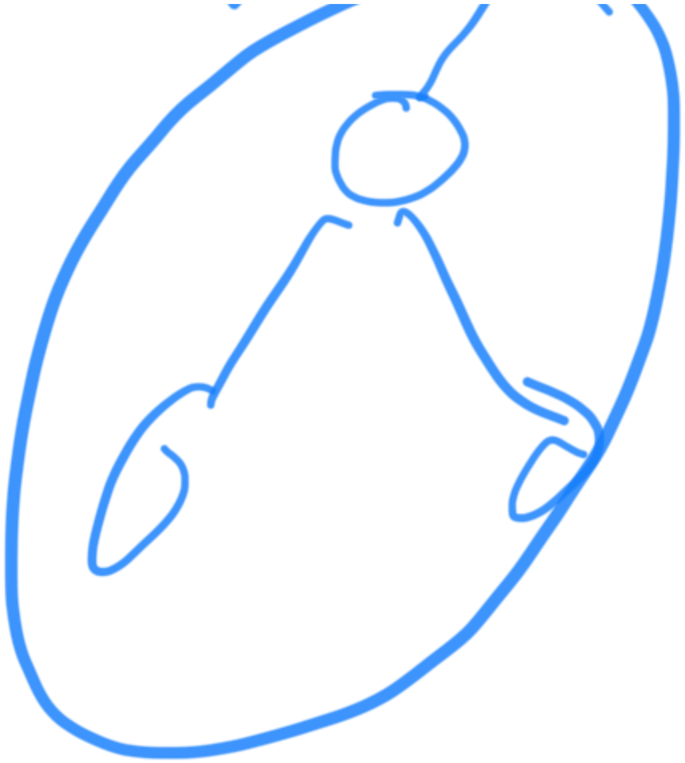
lR

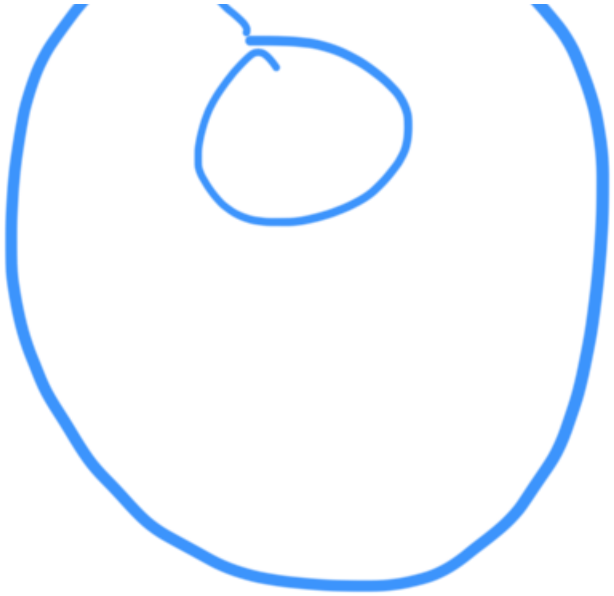
← isValidBST (root.right)

&&

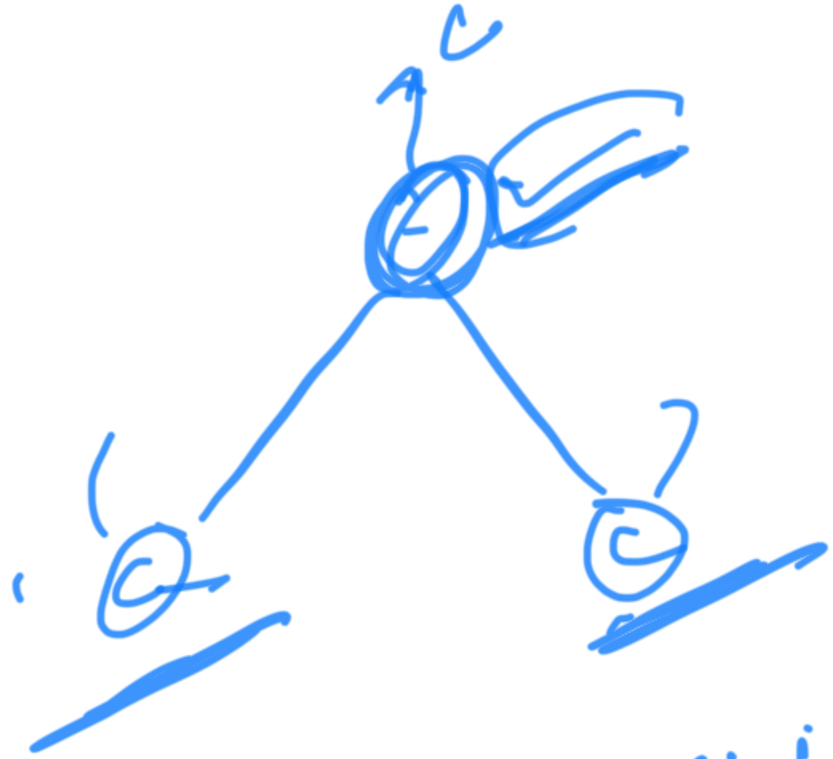
(lr.max < root && rr.min > root)







Need Camara
has Camera
covered



Left Status

(   )

(   )

Right Station

if(left has a camera or right has a camera)