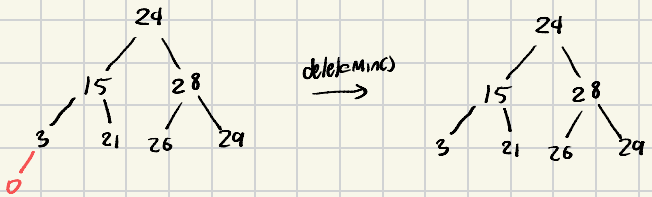
LLRB deleteMin()

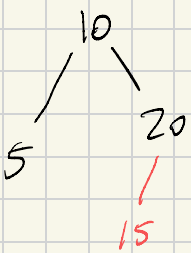
Recall: for LLRB, each path has same # black nodes.

- because of this, red minimum is the easy case



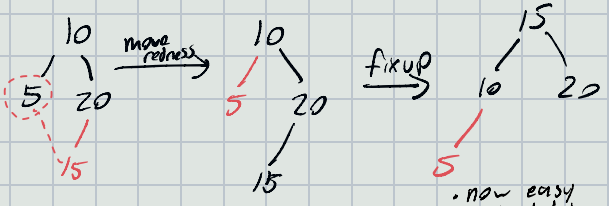
no need to change anything! All paths retain their black nodes.

- Ok, let's do the harder case



deleteMin()?

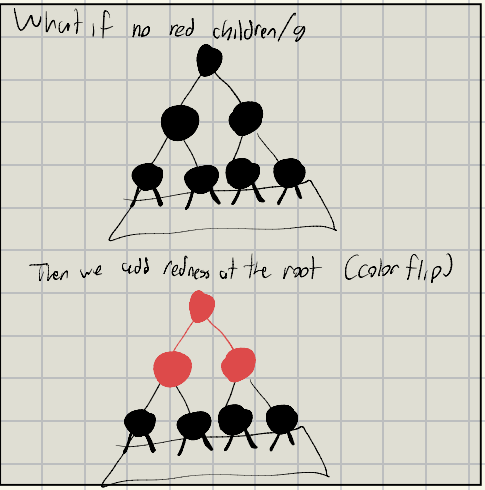
The idea is to move redness to the min element, e.g.,



Algorithm Pseudocode (root r)

1) If we are at the min elt (r.left=null) : delete it.
2) Otherwise,
   i) if either the left child or its left-left grandchild are red: continue recursing (deleteMin(r.left)).
   ii) Otherwise, take the redness from the right-left grandchild then continue recursing
3) At the end, fix the tree from bottom up e.g. running fixup(leaf.parent), fixup(root.parent.parent), ..., fixup(root)
4) Set root = black.

example + function code next page



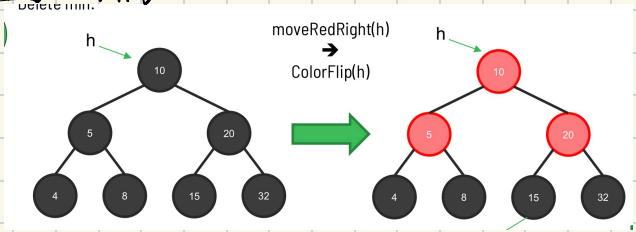
The important code

```
deleteMin(r):
    if r.left = null:
        r = minimum, delete.
    else, if r.left and r.left.left are both black:
        r = moveRedLeft(r)
    r.left = deleteMin(r.left)
    return fixup(r)
// lastly, turn root=black
```

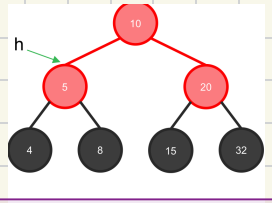
Imagine we are just running fixup up until the root.

```
moveRedLeft(r):
    flipColors(r)
    Flip root color and children color
    if r.right.left = red:
        rotRight(r.right)
        r = rotateLeft(r)
        flipColors(r)
```

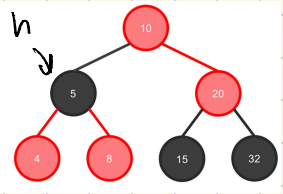
Example: first add redness to 4



deleteMin(5) →

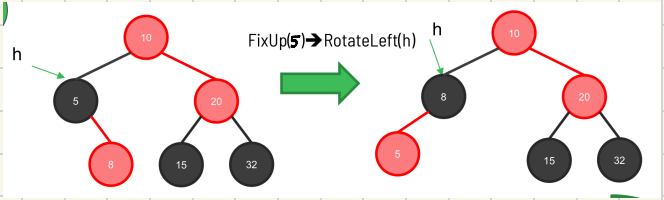


colorflip(5) →

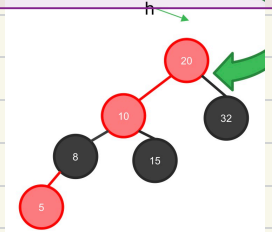


(fixup is the same as in insert).

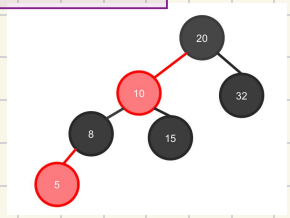
Can now delete 4. Then fixup(5), fixup(10)



fixup(10) →
(rotateLeft(10))



turn root black →



```
function fixUp(h):
    if isRed(h.right) and not isRed(h.left):
        h = rotateLeft(h)
    if isRed(h.left) and isRed(h.left.left):
        h = rotateRight(h)
    if isRed(h.left) and isRed(h.right):
        flipColors(h)
    return h
```