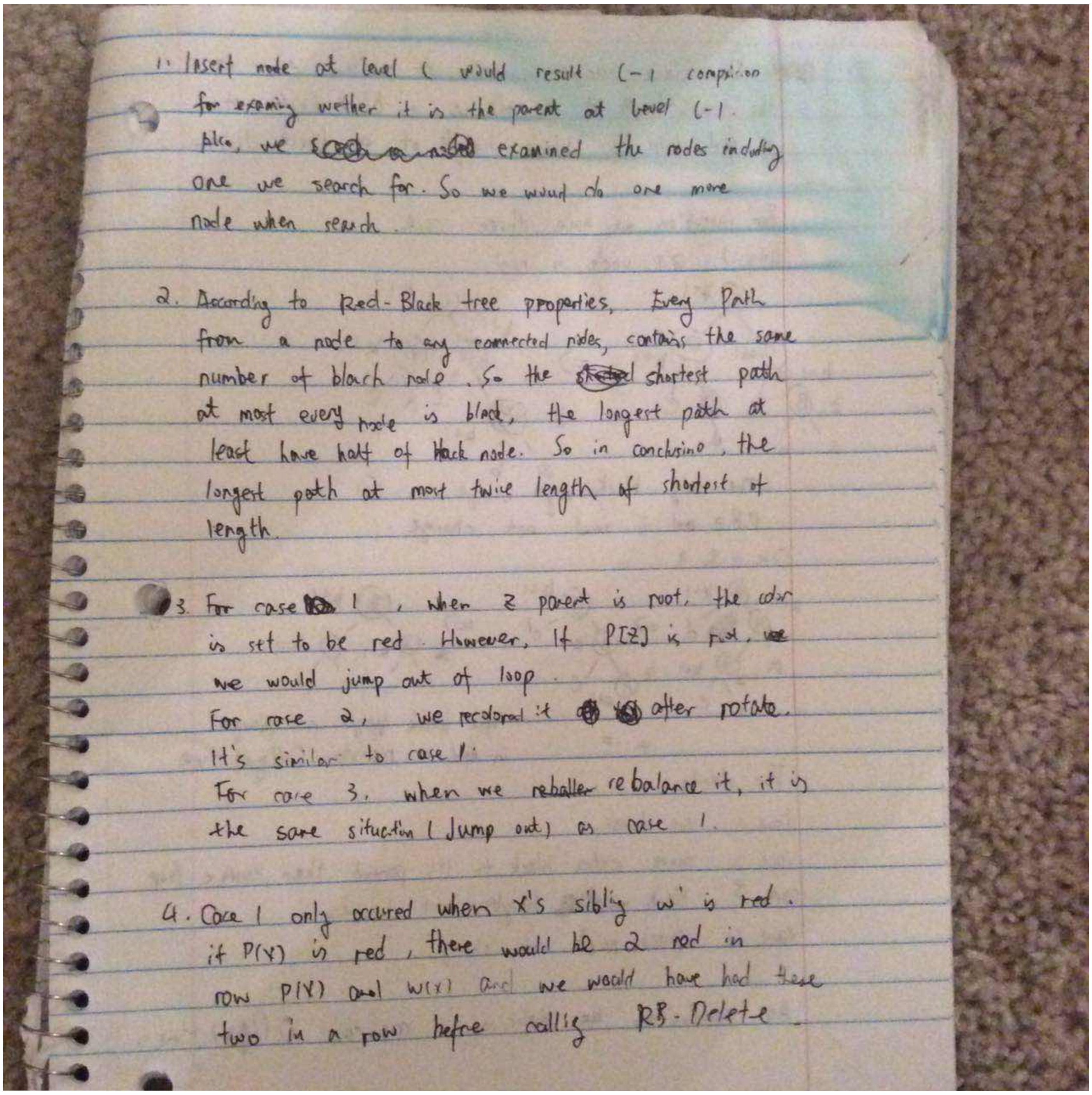


1. Insert node at level l would result l-1 comparision for examing wether it is the parent at level l-1. Also, we [illegible] examined the nodes including one we search for. So we would do one more node when search.

2. According to Red-Black tree properties, Every Path from a node to any connected nodes, contains the same number of black node. So the [illegible] shortest path at most every node is black, the longest path at least have half of black node. So in conclusino, the longest path at most twice length of shortest of length.

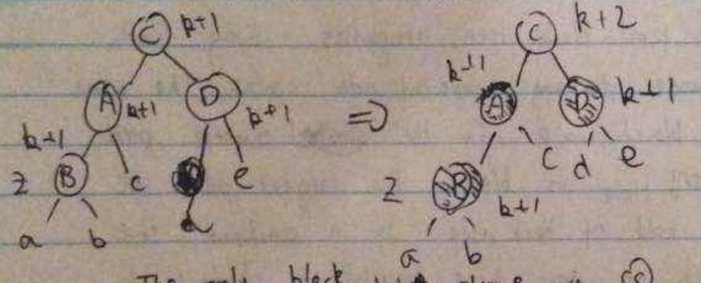
3. For case [illegible] 1, when z parent is root, the color is set to be red. However, if P[z] is root, [illegible] we would jump out of loop.
For case 2, we recolored it [illegible] after rotate. It's similar to case 1.
For case 3, when we ~~reballen~~ rebalance it, it is the same situation (Jump out) as case 1.

4. Case 1 only occured when x's sibling w is red. if P(x) is red, there would be 2 red in row P(x) and w(x) and we would have had these two in a row before calling RB-Delete.

5. The black-weight is better for consider due to the black-height of a node is a black-height of a red child or black-height of a black child.

For insertion, we have three case

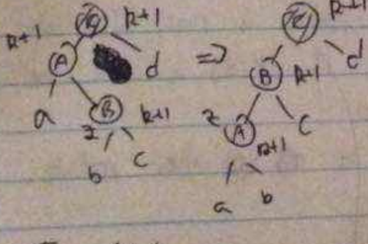
case 1: z's uncle is red

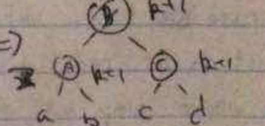


The only black-height change is C

p.p.z and z would not change.

Case 2 & 3





The black height not change in end. maintain O(lgn) time.

For delete

Case 1 Not change.

Case 2 move extra-black to its parent then contine loop

case 3 Black ~~weight~~ height not change

case 4 Black height not change.

As conclusion, the deletion maintain O(lgn) time.