Excersize 1

| | Insert | Delete | (exact) Lookup | (closest) Lookup |
|---|---|---|---|---|
| Array | $\Theta 1$ | $\Theta n$ | $\Theta n$ | $\Theta n$ |
| Linked List | $\Theta 1$ | $\Theta n$ | $\Theta n$ | $\Theta n$ |
| Hash Table | $\Theta 1$ | $\Theta 1$ | $\Theta 1$ | $\Theta n$ |
| Heap | $\Theta(\log n)$ | $\Theta n$ | $\Theta n$ | $\Theta(\log n)$ |

↗ can be $\Theta(\log n)$

↑ not sure

Excersize 2

There is two cases: If the array is sorted/unsorted

→ If not sorted

- Look at every individual element in array
- compare it to $x_q (i - x_q)$
- if the value is smaller than the prev., call it the closest
- do until no more elements

→ If sorted

- Probably use binary search to easily find in between what elements $x_q$ should be in between
- compare both values to $x_q$
- whichever is smallest, is closest

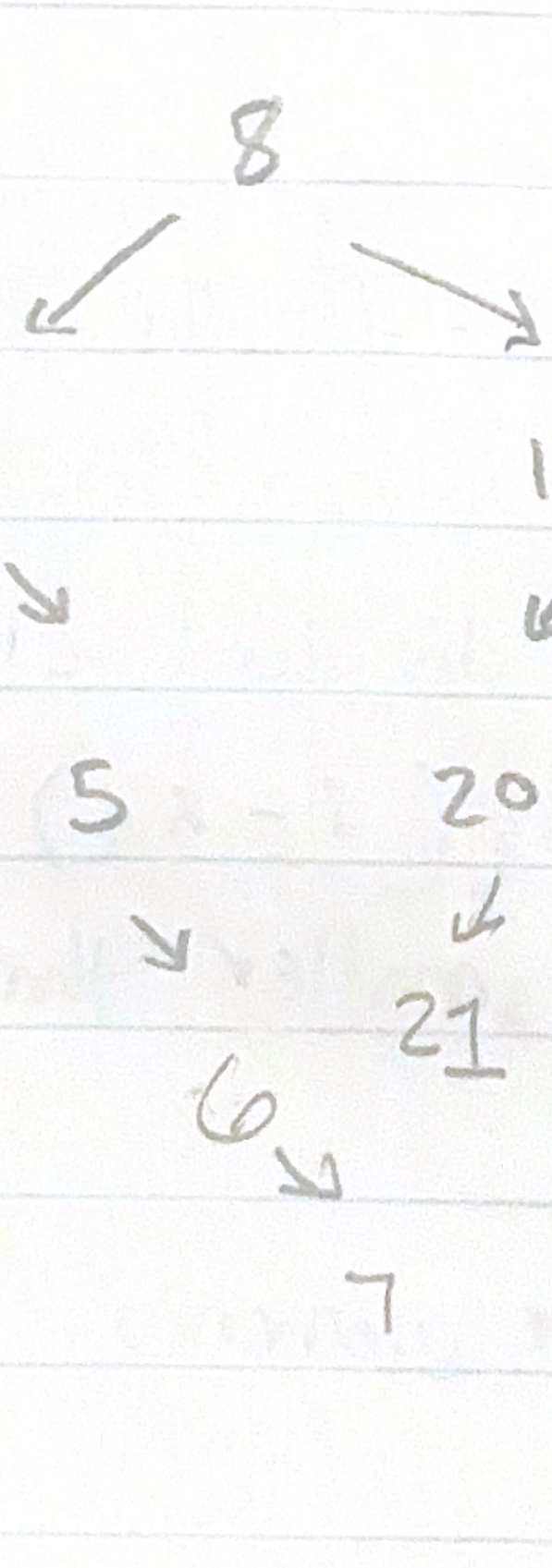
Excersice 3

Since we are using binary search, we know that it would take $\Theta(\log n)$ to find where we want to add/delete

→ add/deleting means elements are all shifted by one either right or left

↳ $\Theta(n)$ time

Excersize 4




Excersice 5

Properties of Red-Black-Tree:

→ Node is red or black

→ ROOTS are black

→ Reds can't stack (be next to eachother)

→ Root to leaf has same # of black nodes

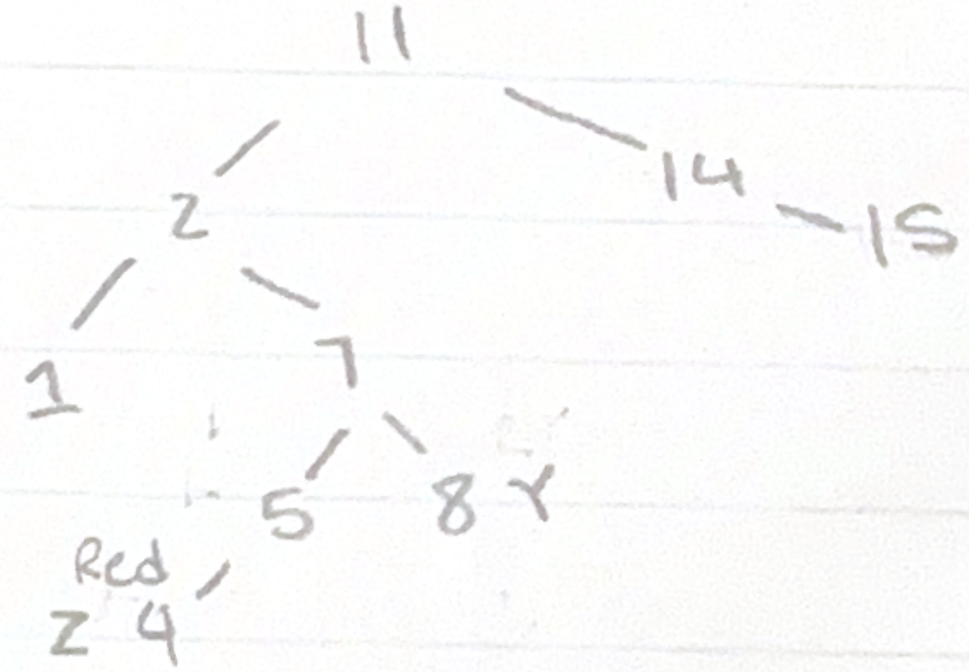
The LONGEST path would be twice the SHORTEST path → the shortest path would just be black nodes

we can maybe say height of tree ≤ 2 × (shortest path)

Had to look at hint...

h.o.t $\leq 2\log_2(n+1)$ ← the black nodes follow this MEANING the worst case would fall under $\Theta(\log n)$

Excersize 6




In case 1: 11, 14, 15 stay...