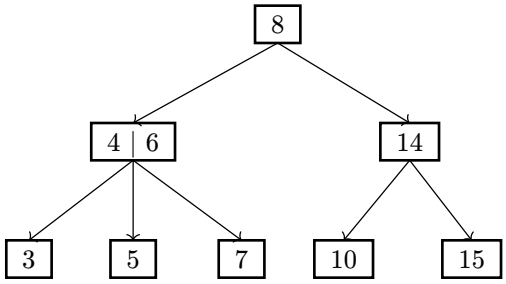
CS 61B
Fall 2025

B-Trees, LLRBs, and Heaps

Discussion 07: October 13, 2025

1 2-3 Trees and LLRBs

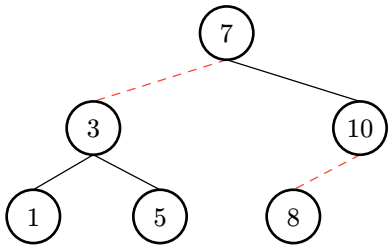
(a) Draw what the following 2-3 tree would look like after inserting 18, 38, 12, 13, and 20.



(b) Now, convert the resulting 2-3 tree to a left-leaning red-black tree.

(c) If a 2-3 tree has depth `H` (that is, the leaves are at distance `H` from the root), what is the maximum number of comparisons done in the corresponding red-black tree to find whether a certain key is present in the tree?

(d) Now, insert 9 into the LLRB Tree below. Describe where you would insert this node, and what balancing operations (`rotateLeft`, `rotateRight`, `colorSwap`) you'd take to balance the tree after insertion. Assume that in the given LLRB, dotted links between nodes are red and solid links between nodes are black.



2 Absolutely Valuable Heaps

(a) Assume that we have a binary min-heap (smallest value on top) data structure called **MinHeap** that has properly implemented the **insert** and **removeMin** methods. Draw the heap and its corresponding array representation after each of the operations below:

```
MinHeap<Character> h = new MinHeap<>();
h.insert('f');
h.insert('h');
h.insert('d');
h.insert('b');
h.insert('c');
h.removeMin();
h.removeMin();
```

(b) Your friendly TA Mihir challenges you to create an integer max-heap without writing a whole new data structure. Can you use your min-heap to mimic the behavior of a max-heap? Specifically, we want to be able to get the largest item in the heap in constant time, and add things to the heap in $\Theta(\log n)$ time, as a normal max heap should.

Hint: You should treat the **MinHeap** as a black box and think about how you should modify the arguments/ return values of the heap functions.

3 Extra: Heap Mystery

We are given the following array representing a min-heap where each letter represents a **unique** number. Assume the root of the min-heap is at index one, i.e. `A` is the root. Our task is to figure out the numeric ordering of the letters. Therefore, there is **no** significance of the alphabetical ordering. i.e. just because B precedes C in the alphabet, we do not know if B is less than or greater than C.

Array: [-, A, B, C, D, E, F, G]

Four unknown operations are then executed on the min-heap. An operation is either a `removeMin` or an `insert`. The resulting state of the min-heap is shown below.

Array: [-, A, E, B, D, X, F, G]

(a) Determine the operations executed and their appropriate order. The first operation has already been filled in for you!

Hint: Which elements are gone? Which elements are newly added? Which elements are removed and then added back?

1. `removeMin()`
2. _____
3. _____
4. _____

(b) Fill in the following comparisons with either >, <, or ? if unknown. We recommend considering which elements were compared to reach the final array.

1. X _____ D
2. X _____ C
3. B _____ C
4. G _____ X