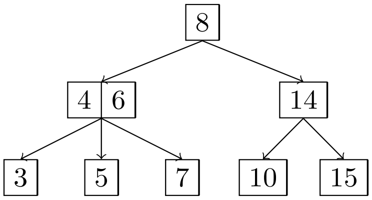
CS 61BL Summer 2021

Balanced Search and Tries

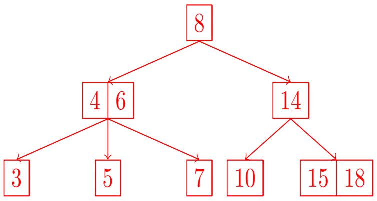
Recurring Section 8: Thursday July 22, 2021

1 2-3 Trees and LLRB's

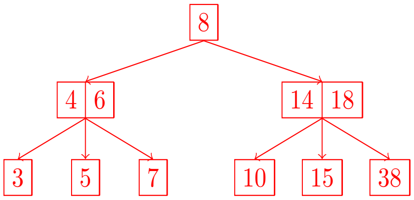
a) Draw what the following 2-3 tree would look like after inserting 18, 38, 12, 13, and 20.



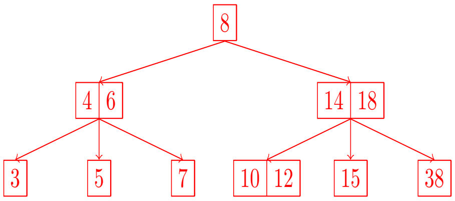
Adding 18:



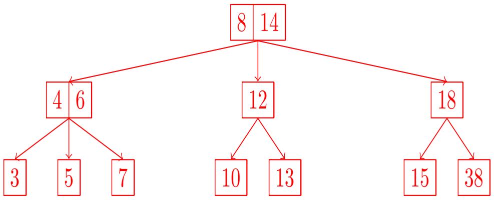
Adding 38:



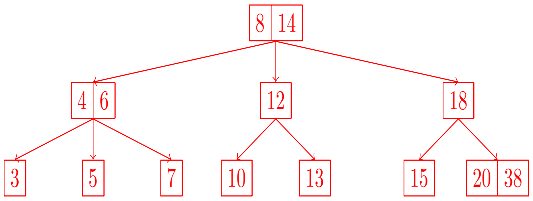
Adding 12:



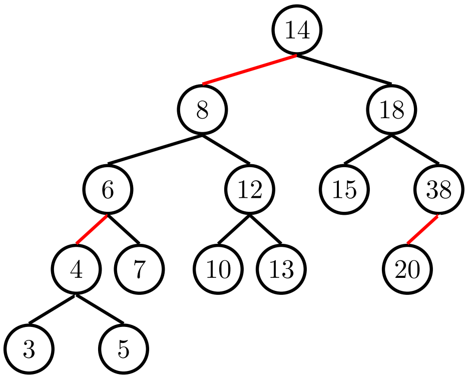
Adding 13:



Adding 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, there are `H` number of edges in the path from leaf to 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?

$2H + 2$ comparisons.

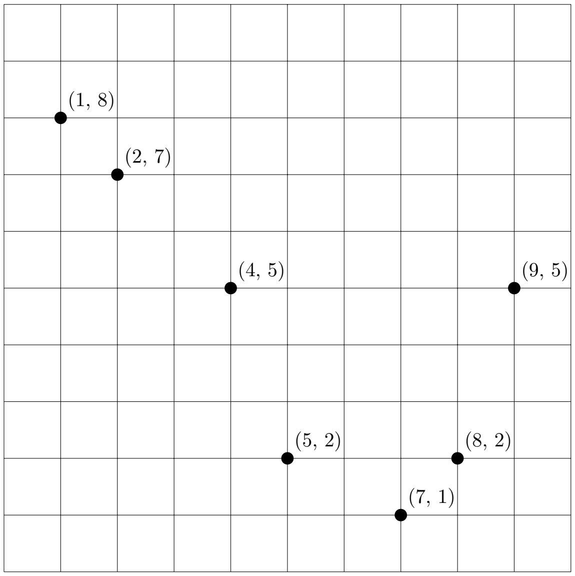
The maximum number of comparisons occur from a root to leaf path with the most nodes. Because the height of the tree is H, we know that there is a path down the leaf-leaning red-black tree that consists of at most H black links, for black links in the left-leaning red-black tree are the links that add to the height of the corresponding 2-3 tree. This means that there are $H + 1$ nodes on the path from the root to the leaf, since there is one less link than nodes,

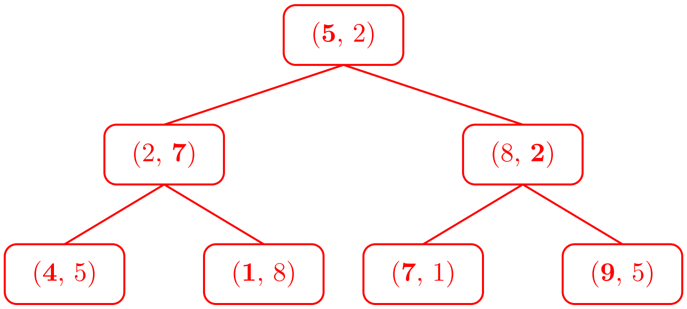
In the worst case, in the 2-3 tree representation, this path can consist entirely of nodes with two items, meaning in the left-leaning red-black tree representation, each blank link is followed by a red link. This doubles the amount of nodes on this path from the root to the leaf.

This example will represent our longest path, which is $2H + 2$ nodes long, meaning we make at most $2H + 2$ comparisons in the left-leaning red-black tree.

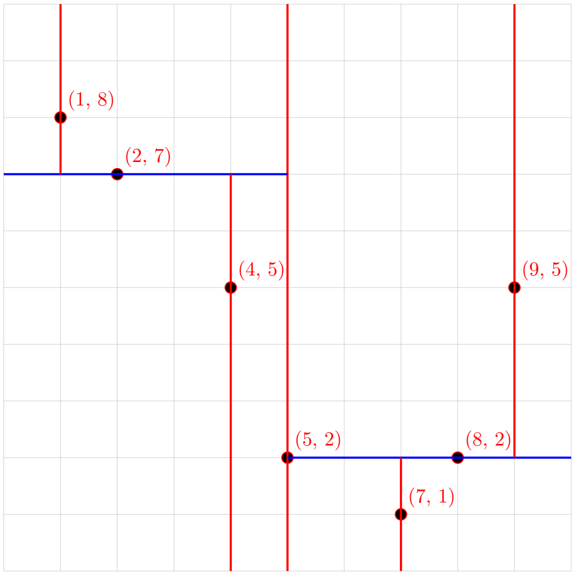
2 KD Trees

Given the points shown in the grid to the right, draw a perfectly balanced k-d tree. For this tree, first split on the x dimension. The resulting tree should be complete with height 2. Then, draw the corresponding splitting planes on the grid.



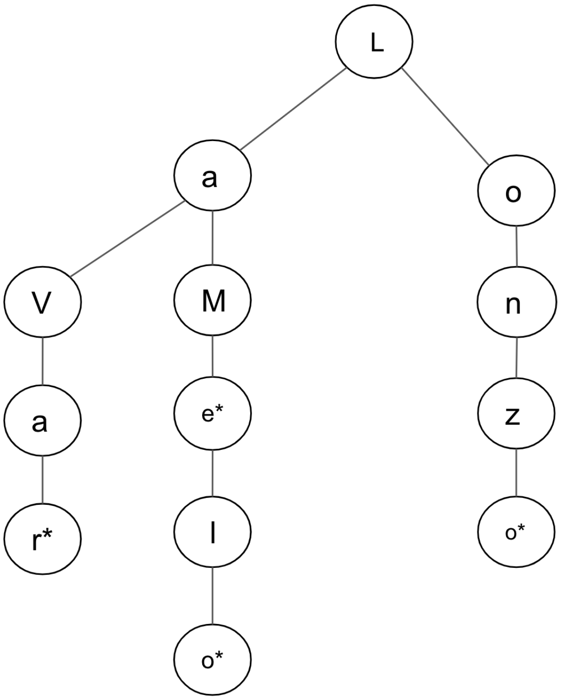


We can also visualize the k-d tree points in space by drawing on the splitting planes as is shown below.



3 Trie Me

The Big Baller Brand has decided to use a trie to have fast lookup of their Big Ballers. Currently, the state of the trie is as follows:

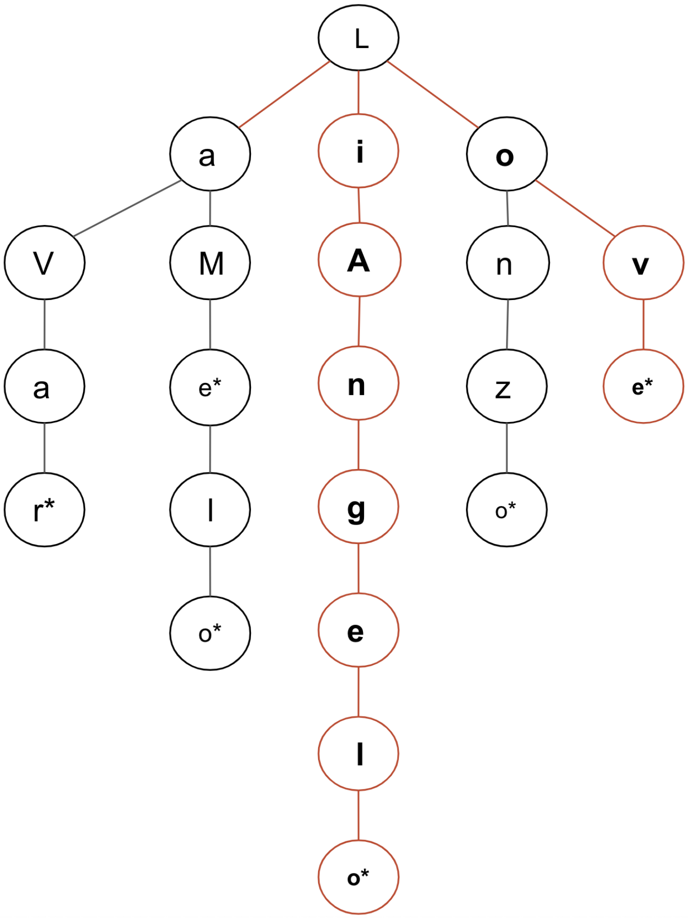


1. The Biggest Baller of them all, CEO LaVar Ball, enjoys being reminded of who is a Big Baller. Remind him of who the Big Ballers are by finding all the words in the trie. Note: The nodes with an asterisk denote the end of a word.

   The words are: LaVar, Lame, LaMelo, Lonzo

2. Not again! LaVar Ball has forgotten about his son LiAngelo once again. Help LaVar by inserting ”LiAngelo” and ”Love” into the trie above so that no Big Baller is forgotten.

   The trie after inserting ”LiAngelo” and ”Love”:



3. How long does it take to add n words, each of max length L?

   It takes $O(nL)$ time, since we take $O(L)$ time to add one word, and we add n words.

4. What's the best and worst case runtime to check whether a word of length L is in the trie?

   Best case $\Theta(1)$: the first letter of the word is not in the trie.

   Worst case: $\Theta(L)$: the word is in the trie, and we have to traverse to the end of the word to confirm.