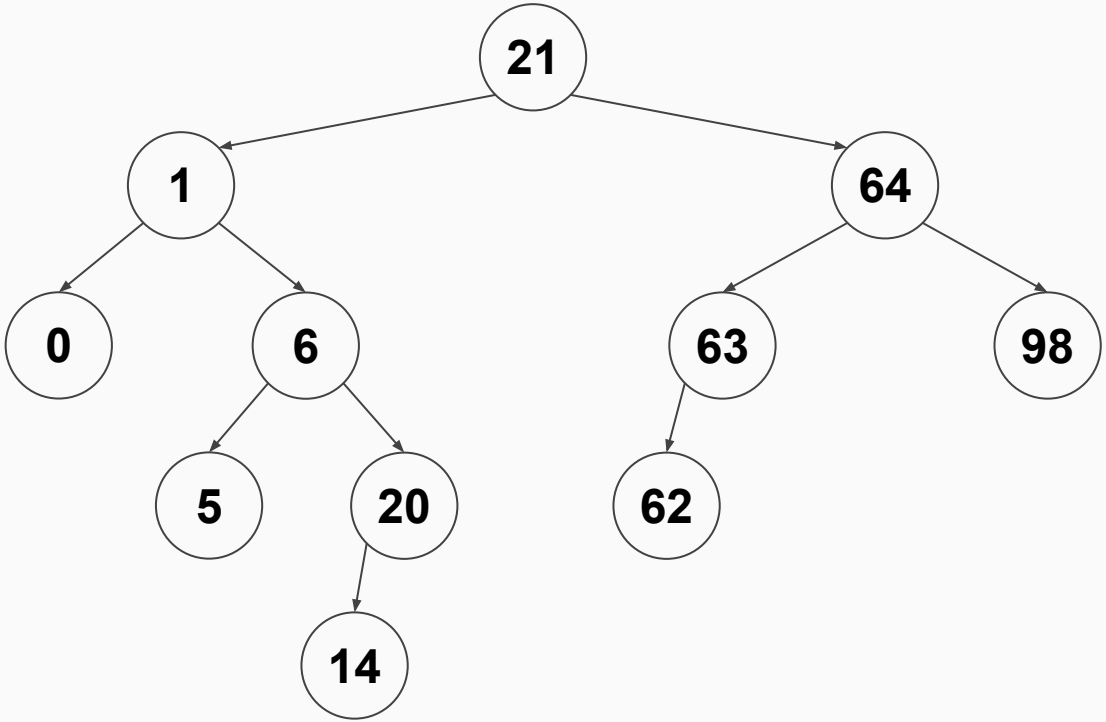
CSE 250 Recitation

April 30 - May 1: Final Review

Discussion

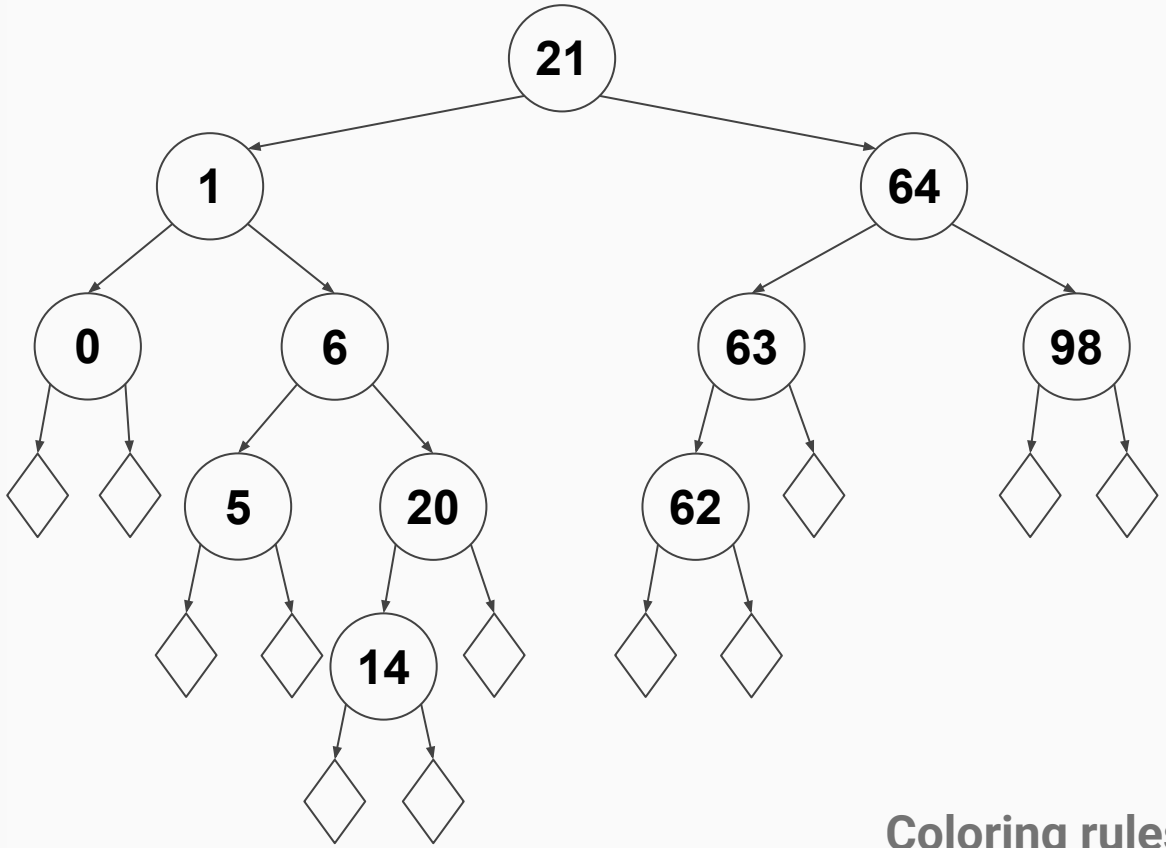
How can we determine if this tree meets Red-Black tree constraints?



Exercise #1

Check if depth of the deepest empty node is at most 2x the shallowest for EVERY subtree.

Equivalently, see if the tree can be colored.



ALWAYS DRAW YOUR EMPTY TREE NODES!

Coloring rules:

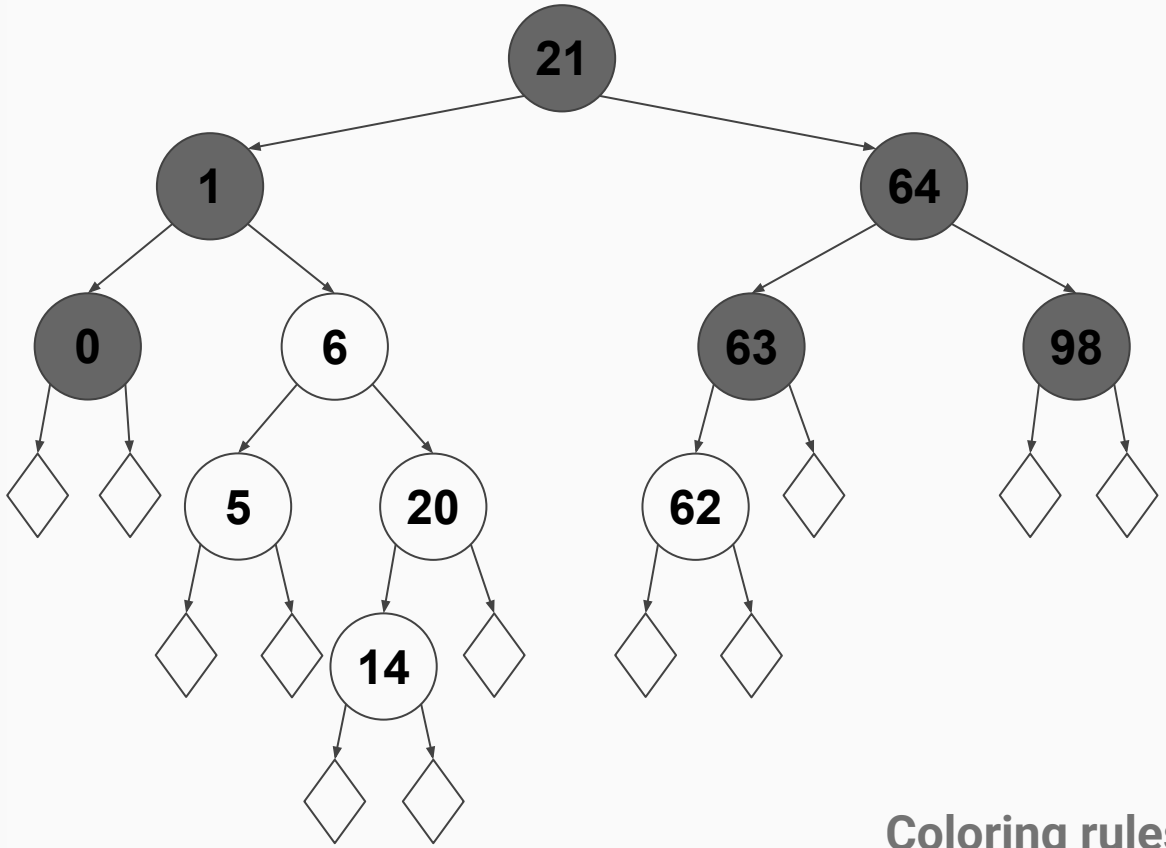
Equal # of black nodes from empty node to root

No red-red connections

Exercise #1

Check if depth of the deepest empty node is at most 2x the shallowest for EVERY subtree. **IT IS!**

Equivalently, see if the tree can be colored.



ALWAYS DRAW YOUR EMPTY TREE NODES!

Coloring rules:

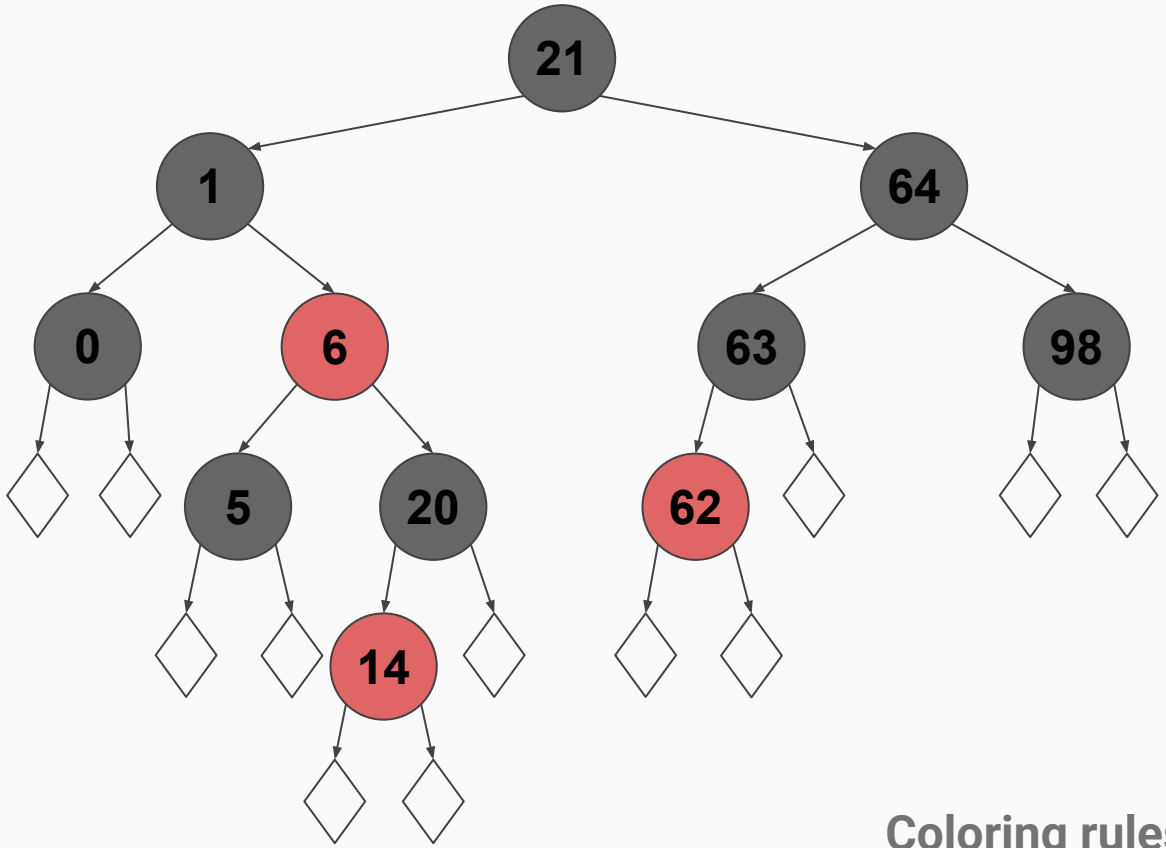
Equal # of black nodes from empty node to root

No red-red connections

Exercise #1

Check if depth of the deepest empty node is at most 2x the shallowest for EVERY subtree. **IT IS!**

Equivalently, see if the tree can be colored. **IT CAN BE!**



ALWAYS DRAW YOUR EMPTY TREE NODES!

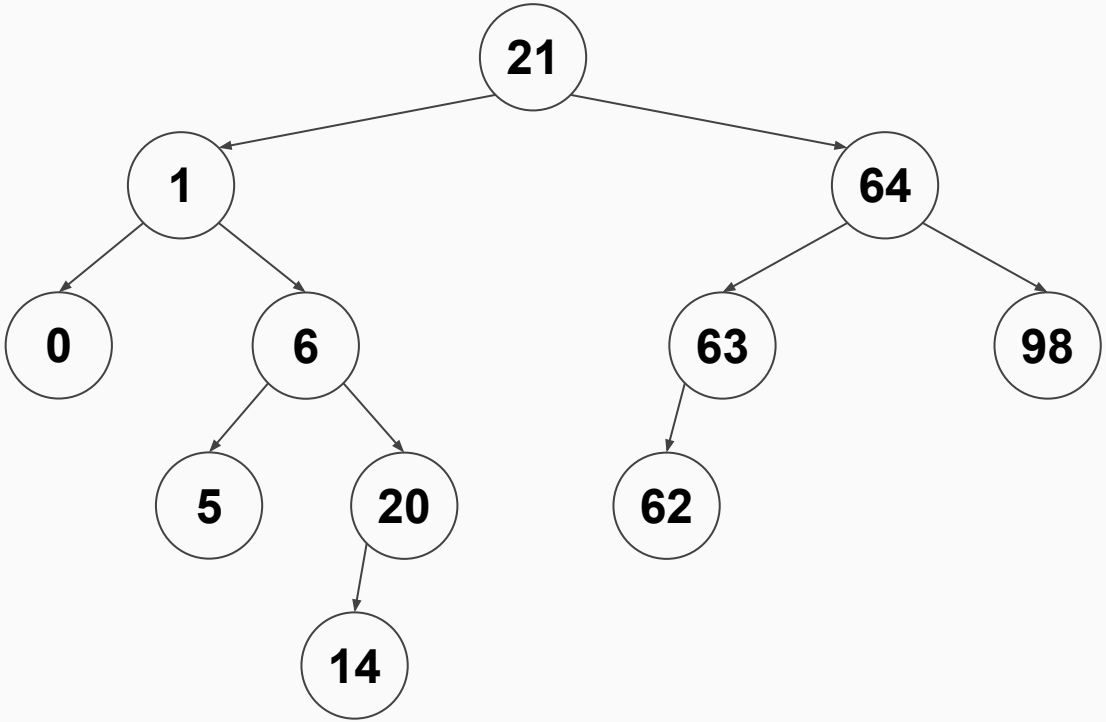
Coloring rules:

Equal # of black nodes from empty node to root

No red-red connections

Discussion

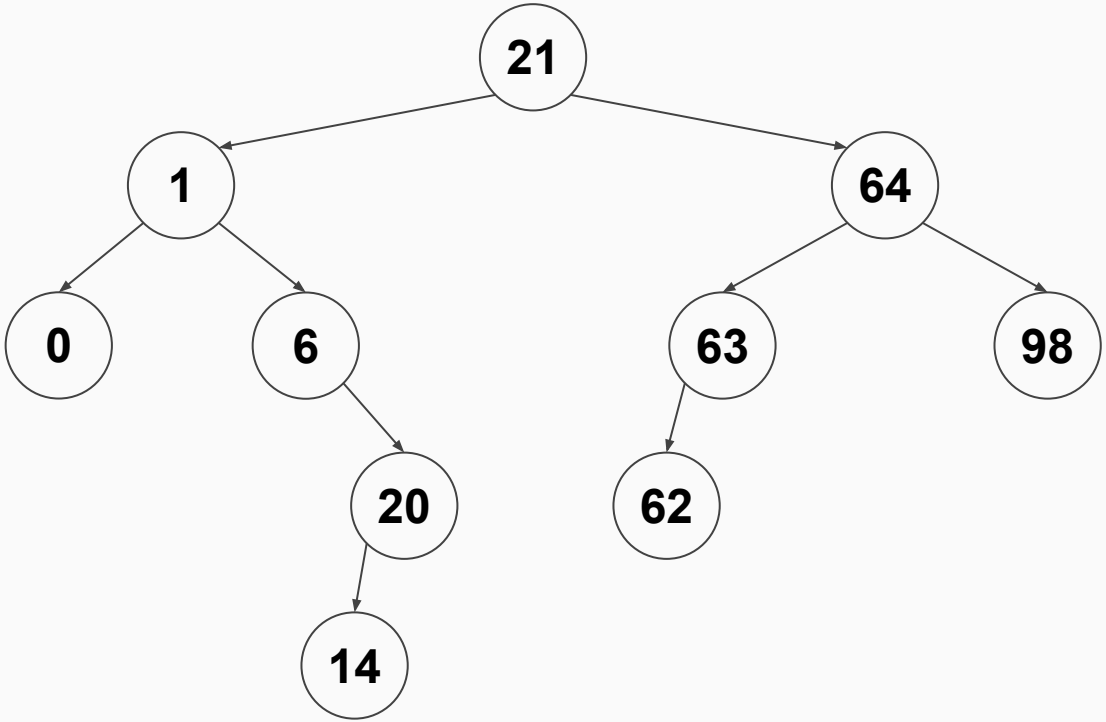
What if we remove the 5?



Discussion

What if we remove the 5?

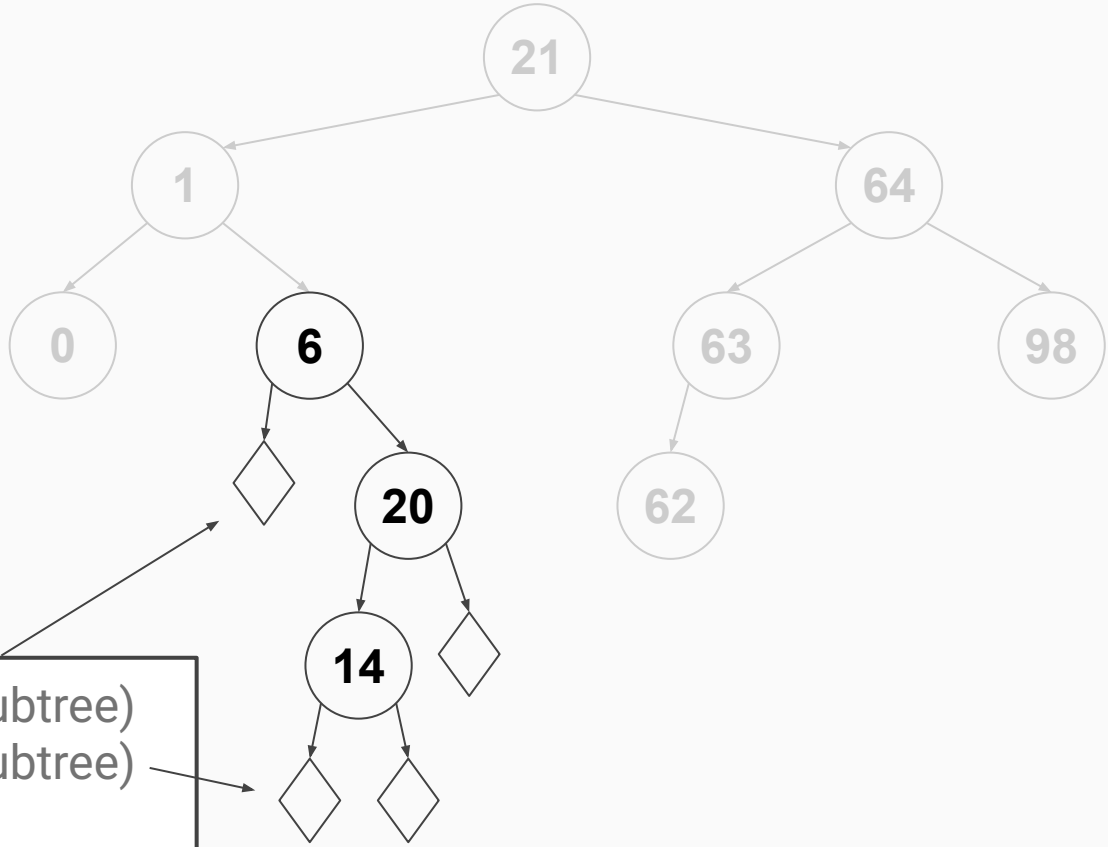
Could this still be Red-Black?



Discussion

What if we remove the 5?

Could this still be Red-Black? **NO!**



Depth 1 (in this subtree)
Depth 3 (in this subtree)
$3 \not\leq 1 \cdot 2$

Remember: All subtrees must also be Red-Black trees as well!

CSE 250 Matrix of ~~Doom~~ Fun

Roll a d6 twice:
https://g.co/kgs/1WcoMvv

The first roll tells you your Data Structure (row)

The second roll tells you your ADT (column)
(if you roll a 6 you get to choose the ADT)

Come up with an implementation for ADT using the Data Structure. Determine the runtime of each key method.

	List	**Stack / Queue**	**Priority Queue**	**Set/Bag**	**Map**
`ArrayList`	Lecture	Lecture		Lecture	
`LinkedList`	Lecture	Lecture	Lecture	Lecture	
`Heap`			Lecture		
`General BST`				Lecture	
`Balanced BST`				Lecture	
`Hash Table`				Lecture	Lecture

CSE 250 Matrix of ~~Doom~~ Fun

Roll a d6 twice:
https://g.co/kgs/1WcoMvv

The first roll tells you your Data Structure (row)

The second roll tells you your ADT (column)
(if you roll a 6 you get to choose the ADT)

Come up with an implementation for ADT using the Data Structure. Determine the runtime of each key method.

	List	Stack / Queue	Priority Queue	Set/Bag	Map
`ArraryList`	Lecture	Lecture		Lecture	
`LinkedList`	Lecture	Lecture	Lecture	Lecture	**TODAY!**
`Heap`			Lecture		
`General BST`				Lecture	
`Balanced BST`				Lecture	
`Hash Table`				Lecture	Lecture

Implementing a `Map` with a `LinkedList`

Discussion: What type of data should the `LinkedList` hold?

```
LinkedList <          > data;

V put(K key, V value):


V get(K key):


V remove(K key):
```

Implementing a `Map` with a `LinkedList`

Discussion: What type of data should the `LinkedList` hold?

Exercise: Implement the remaining `Map` methods

```
LinkedList <Pair<K,V>> data;

V put(K key, V value):


V get(K key):


V remove(K key):
```

Implementing a `Map` with a `LinkedList`

Discussion: What type of data should the `LinkedList` hold?

Exercise: Implement the remaining `Map` methods

```
LinkedList <Pair<K,V>> data;

V put(K key, V value):
    old = remove(key)
    data.add(new Pair(key,value))
    return old

V get(K key):
    for pair in data:
        if pair.key == key:
            return pair.value
    return null

V remove(K key):
    for pair in data:
        if pair.key == key:
            data.remove(pair)
            return pair.value
    return null
```

Blooket Review

https://dashboard.blooket.com/set/6627c6c3a688259d39444174