For BSTs, the most inefficient way to add is in to put it in order.

In a 2-3 tree, what is the worst case insertion order?

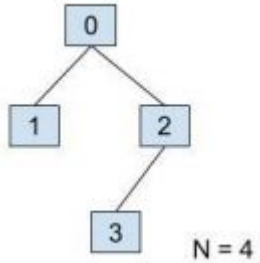
- What is the worst case?
  - A tall tree. A tree with great height.
- Let's think about in order insertion.
  - If we insert 7 items in order, what tree do we get? 1, 2,3 , 4, 5, 6, 7
- I"m tabling this one because I can't draw fast enough without a pen/

Talk about nodes in a B-tree being half full?

- Out of scope

Worst case height of a weighted quick union:

- How is that dependent on number of nodes?
- The worst case way to get to height two trees is to union two worst case height 1 trees, giving you this shape ----->
- What is smallest N that gives height 3 tree?
  - And how does it work?
  - Merge two minimum sized height 2 trees.



How to approach give a task to do, and figure out what data structure to use?

- Usually with some runtime constraint.

First what tools do we have?

- HashMaps → what are they good for? Mapping, and how long do they take in the worst case for a get/put operation? Theta(N). If things are nicely distributed though, you get Theta(1).
  - Why not just use a TreeMap which is guaranteed Theta(log N)?
  - Items may not be comparable AND generally a little tiny bit faster in practice.
- Priority Queue: Lets you maintain a dynamic collection of items sorted by some value.

Read in a text file and print all of the unique words that appear. The words should be printed in alphabetical order

- We need to find all the UNIQUE words [did we mean words that occur only once?] Let's say no:
  - Let's try a set.

Use a TreeSet<String>

- For each word in the file, insert into the set.
- For each word in s.keySet() print it out. < --- will this be alphabetical order? Actually yes, because TreeSet maintains keys in sorted order.
- But if you're not sure if treeset has keys in sorted order, what can you do? Remove them all, put them in an array, use selection sort or mergesort. OR put them in a priority queue, delete them out one by one.

Read in a text file and print all of the unique words that appear. The words should be printed in alphabetical order

- We need to find all the UNIQUE words [did we mean words that occur only once?] Let's say yes.
  - Let's try a set.

Use a TreeMap<String, Integer>

-

Erweiterten netzwerk:

- WeightedQuickuNion to track connectness

Map<String, Integer> → username

HashMap vs. **TreeMap**? We need O(log N) time.

- There is a chance (however small) that all the usernames end up going to the same bucket.

Erweiterten netzwerk:

- WeightedQuickuNion to track connectness

Map<String, Integer> → username

HashMap vs. **TreeMap**? We need O(log N) time.

- There is a chance (however small) that all the usernames end up going to the same bucket.

Let’s talk about LLRB height.

- Theta height first.
- We do not have a procedure for adding to an LLRB directly (see optional slides).
- They are isometric to a 2-3 tree.
- If I have a 2-3 tree of height Q, what is the worst case height of its corresponding LLRB?
  - 2Q: How could this happen? Alternating red/black links on left spine.
  - The worst case for a 2-3 is theta(log N) (see class), so worst case is theta(2 log N) = theta(log N)
-

Come up with a sequence of union and connected operations such that the overall runtime of the sequence is O(N + #Union + #connected).

- Include the runtime to create the WQU object (since this problem is building on problem 3, not shown).

```
doStuff() {
  WQU x = new WQU(x);                    // runtime is theta(N)
  x.union( … ); do this lots of times        // each needs to be constant
  x.connected( … ); do this lots of times // each needs to be constant
}
```

Why is the answer “don’t do any union or connected calls at all” bad?

With clever insight, we realized the problem is really

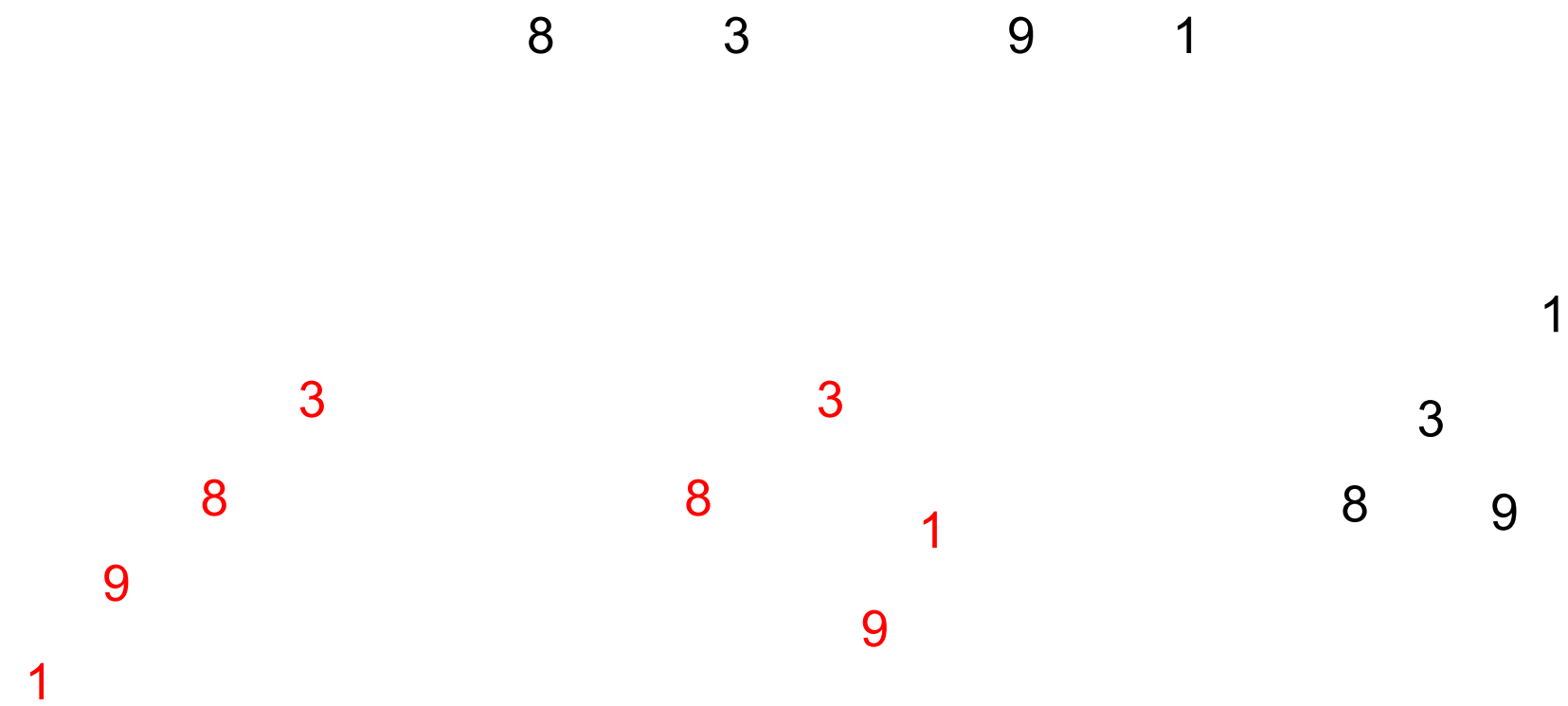
- GIve a bunch of union and connected calls such that they are all constant time.
- Note: THe runtime of these ops is the runtime of find.
  - What is the runtime of find? THe height of three.

We want to find a sequence of operations such that the height of the tree is … ALSO CONSTANT.

- examples : union(i, i); a bunch of times [trees of height 0]
- Examples: union(0, i); a bunch of times [tree of height 1]

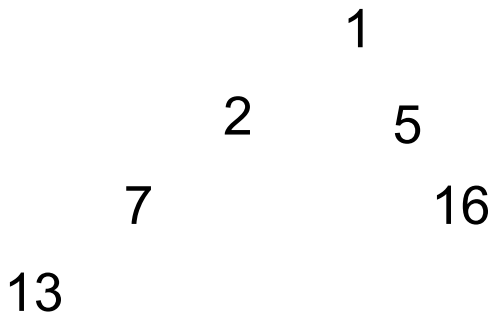
Prove that when you do range finding it takes theta(log N + R) where R is the number.

- A little too scared to do this in this room right now see piazza.
-



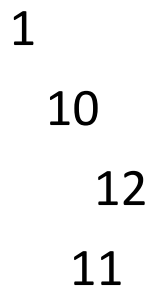
Find the smallest item, put it in the root

- Then “insert” each item.
- Try to put the item in the bottom left , sliding it up the tree -- breaks for 8 because it goes in the wrong place



Adding elements to a binary search tree.

- 1, then 10, then 12, then 11



WRiting godo hashcodes: You should understand why the intiial ahscode from lecture wer enot good (multiplying by powers of 32)

If I do a 2-4, can I end up with one item in every node, yes, but I’m not sure of the seuqence.

For hash tables, do you insert at the end or beginning of list?

- Doesn’t matter, the important point is that you make sure not to have duplicates.

Data structures questions: OK to assume hashCode is well written? YES, but well writen is no guarantee of performance because of th pigeonhole performance.

\

Find the smallest item, put it in the root

- Then “insert” each item.
- Try to put the item in the bottom left , sliding it up the tree -- breaks for 8 because it goes in the wrong place