Announcements

- This is our last lecture

- TA hours end on Friday (Nick will post some hours on calendar for next week)

- Final exam review Q&A on Wednesday (12/13) 2pm (right here); prep guide online soon

- Check your remaining late days for HW6

- Look for announcement on grade reports

Scenario 1: Online Listings for Room Rentals

Renting out rooms for a weekend convention

- Each listing contains the rental price for the weekend, the number of beds the unit has, the street address, and an optional photo. The site also stores the email address of the person who posted each listing (email addresses won't be displayed on the site)
- When a renter visits the site, they can search on either or both of the **number of beds** or the **price range**. The site displays those listings that match the search requirements.
- Listings get added to or removed from the site throughout the time that the site is up.

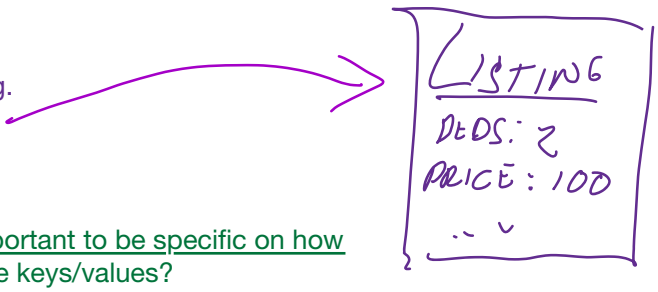
Questions

Which actions occur frequently, and thus need efficient run-time?

What data structure(s) do you propose? Include types of elements/keys/etc

Note: these problems are a bit harder than we would ask on the final—we are using more challenging problems here so we have more to discuss!

First, Need a way to represent each listing (eg. object with price, beds, etc)
=> how to store efficiently to search?



When discussing data structure choices: important to be specific on how data is used. Eg. if using hashmaps, what are keys/values?

OPTIONS:

HASHMAP < PRICE, LIST OF LISTINGS >

HASHMAP < #BEDS, LIST OF LISTINGS >

ONE OR BOTH?

MAP< PRICE, MAP<#BEDS, LIST OF LISTINGS >>

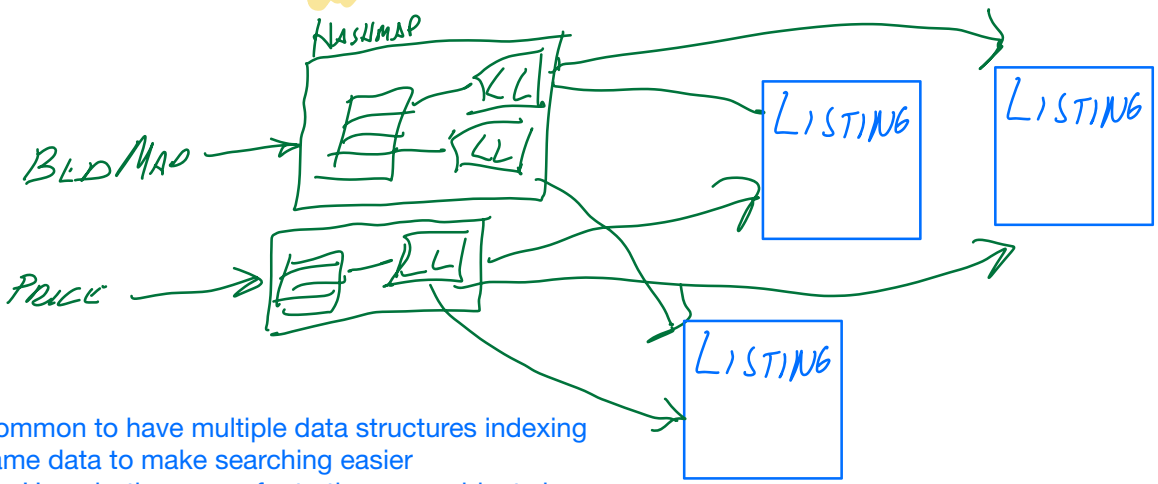
Could have nested hashmaps, but would be hard to search if you only know one parameter

---

On having two hash maps…

- HASH MAP < #BEDS, LIST< LISTING >>

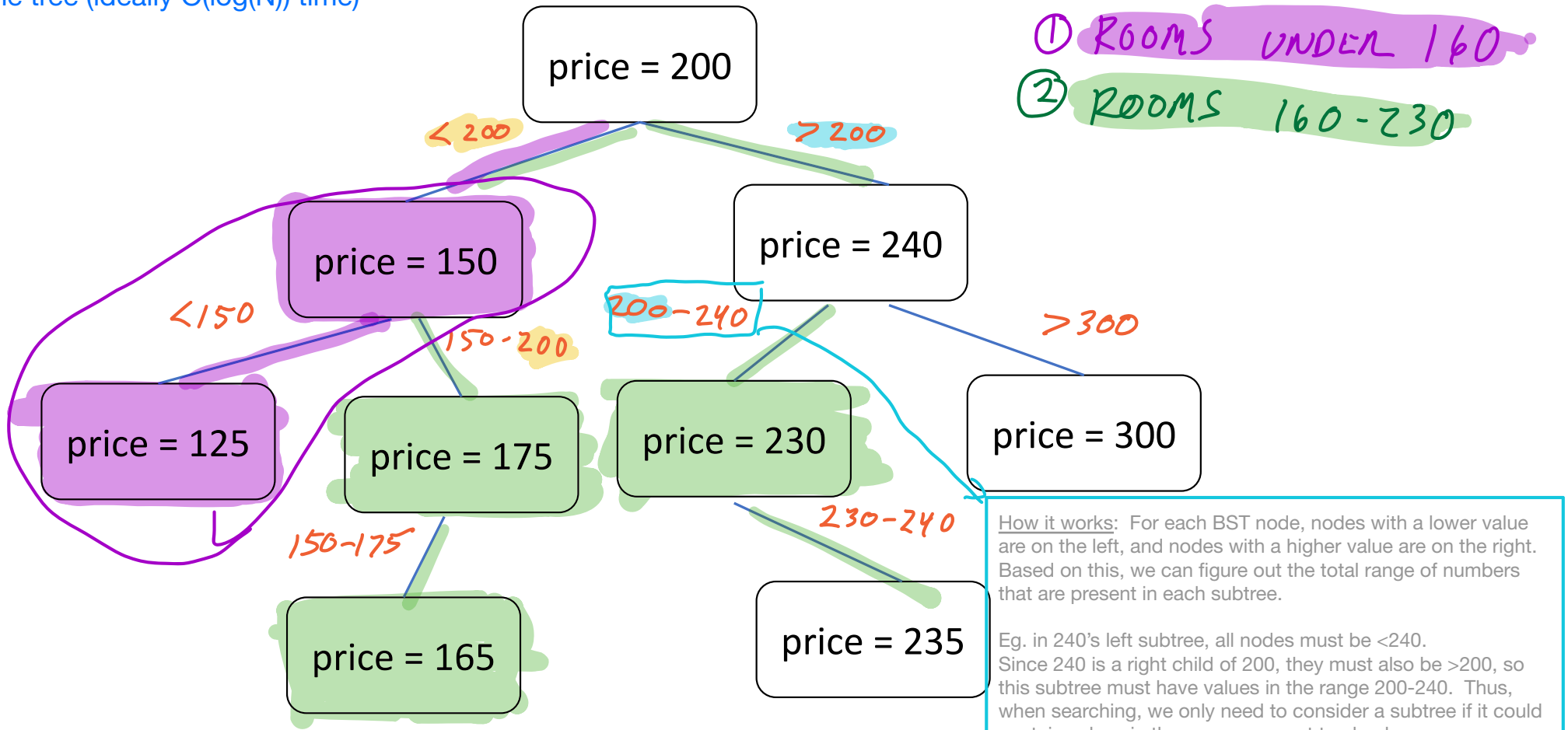
HASH MAP < PRICE RANGE, LIST< LISTING >>



Common to have multiple data structures indexing same data to make searching easier
=> Here, both maps refer to the same objects in memory, so this doesn't require extra space (need to keep both maps updated, though)

How to search over different ranges of prices?

Could use a BST => ordering means we only need to traverse part of the tree (ideally O(log(N)) time)



EXAMPLES

1. ROOMS UNDER 160
2. ROOMS 160 - 230

How it works: For each BST node, nodes with a lower value are on the left, and nodes with a higher value are on the right. Based on this, we can figure out the total range of numbers that are present in each subtree.

Eg. in 240's left subtree, all nodes must be <240. Since 240 is a right child of 200, they must also be >200, so this subtree must have values in the range 200-240. Thus, when searching, we only need to consider a subtree if it could contain values in the range we want to check.

Note: this is quite subtle and more than we would ask on the final.

Scenario 2: Making a Simple Web-based Document Editor

You've decided to create a web-based document editor (like a simplified Google Docs). You'll focus on four operations: adding text, deleting text, undoing edits, and searching for words in the document. The editor should allow documents to have basic styling, such as boldface words/phrases and section headings.

In particular, searching for words needs to be fast (because you imagine people writing large documents in your tool once it becomes popular).

Questions

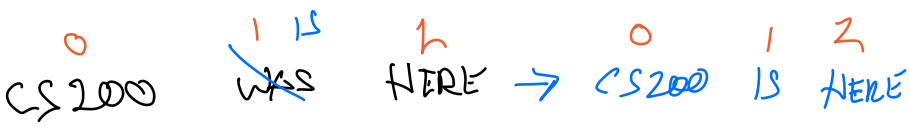
What data do you need to manage for this problem?

What data structure(s) do you propose? Include types of elements/keys/etc

Need to store
- Documents
- Edits/info to enable undo

Two options for document
- One long string => not good for searching
- List of words

In general, how do we capture edits? What is an edit?



EDIT: REPLACED WORD[1] WITH "WAS"

Common practice to store edits, not changed data structures
=> Functional programming vs. notation (remember immutable lists?)
=> Version control systems like git

This is an important in distributed systems, where computations are performed by many unreliable machines running in parallel
=> CS1380: Distributed Systems

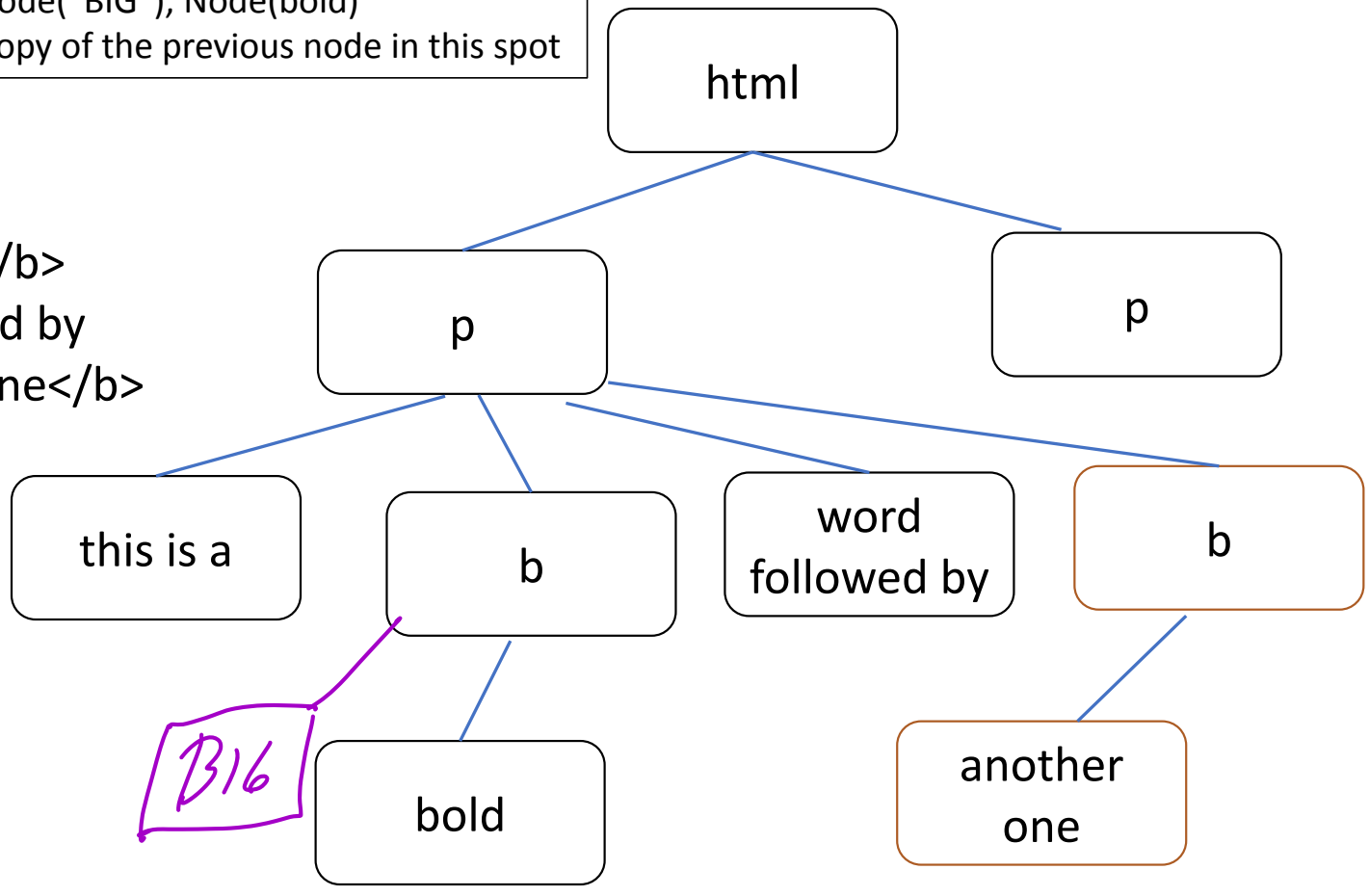
How to store formatting?
Consider: Wikis / HTML / XML

```
<html>
  <p>
    This is a
    <b>bold</b>
    word, followed by
    <b>another one</b>
  </p>
</html>
```

=> Structured way to look at data
If you look hard at it…

Node class:
contents: tag-b
children: Node("BIG"), Node(bold)
previous: copy of the previous node in this spot

```
<html>
  <p>
    This is a
    <b>BIG bold</b>
    word, followed by
    <b>another one</b>
  </p>
  <p>
    new para
  <p>
</html>
```



Things to think about as you leave CS200:

1. **Runtime isn't everything.** Sometimes a simpler data structure that will create fewer bugs is better than doing something more complicated

2. **Code is just one part of writing software** (tests, planning, docs, analysis on how it works) => all of this is just as important, esp. with ChatGPT

3. **You will have bugs.** This is normal, we all do it. Don't think of bugs as something that says you don't belong in CS—it's just part of what we do

4. **Test as you go.** As you write small bits of code, try to test it. Write code, watch it break, do a test. Assignments have tried to walk you through this => try to do it on your own, it will help!

5. **Problems solve you:** working on a problem changes you and how you understand the world => goal is to improve your learning for future problems. If you don't get something 100%, fine, get to 85-90% and move on.

6. **You belong here.** Don't let yourself talk yourself out of it. If you're enjoying CS, stick with it. Don't feel like you get it at this point—I didn't feel like I understood some of this stuff until I started teaching it

Thank you for a great semester!!!

Please stay in touch!