Monday, December 5, 2022 1:42 PM

Scenario #1: A convention is coming to town, and local residents want to make some money by renting out all or part of their homes. In true do-it-yourself mode, they decide to build an online listings system from scratch. You've been hired to plan the data structures for the listings. 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, but the 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.

Question 1: Which problem actions occur frequently, and thus need an efficient run-time?

- search by price range
- search by #beds

pre-sorting the data once isn't the frequent action, but might help the frequent action

want mutability in the "container": quick/easy add/remove

Question 2: What data structure(s) do you propose for the listings? Be sure to include details such as the type of elements in lists, the types of keys/values in hashmaps, etc. Justify your data structure choice, in light of Question 1

How do we store a listing itself?

- As a dictionary? Keys are the "features" of a listing (price, email, #beds, address), corresponding values
- As a class? Fields are price/email/beds/address
  - One of the uses of classes is to group information for a specific "thing" or "concept" or "entity" together

→ when we already have a listing, can look up info in constant time

Want fast search on beds, price range
Dictionary? Something sorted?

1 bed → [. . . .]
3 beds → [. . . .]

Might have two dictionaries:
- # of beds to list of listings
- Price to list of listings

Searching for both at once?
- # of beds to sub-dictionaries that go from price to listing

A dictionary from #beds to _____
A sorted list of listings by price

(L11, $150) linear time to add listing

3 beds → [(L1, $100), (L6, $250)]
1 bed → [(L7, $80), (L4, $80), (L9, $85)]

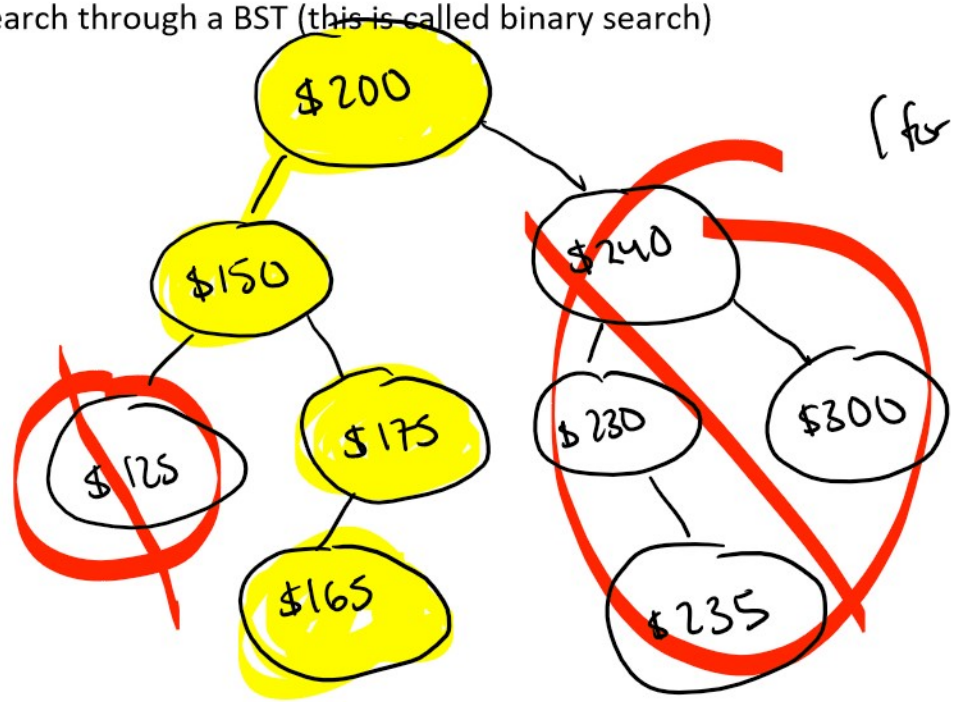
How would we set up this data structure?

1. Go through each listing and put it in the list that sits at the spot in the dictionary for that # of beds (wouldn't worry about sorting)
2. Go through all of those lists and sort by price

Alternative: sort the lists as we go

How would we maintain it?
Let's say we're adding L11 with 3 beds and $150 cost
Constant to look up by # of beds, linear to insert into sorted list

Another solution: instead of sorted list, keep BST at each dictionary entry -- or treat search through list as a search through a BST (this is called binary search)



(for listings $150-$240)

as a list. (have the search each time, just like for a tree!)