Week 10: Maps and Binary Search

Friday, November 13, 2020 9:58 AM

- tables w/ keys & values
  - generic
- look up values based on keys
- keys must be unique
- Operations:
  - get(Object key)
    - returns value associated w/ key
  - put(K key, V value)
    - if key in map:
      - updates value
      - returns prev value
    - if key not in map:
      - adds key/value to map
      - returns null
  - containsKey(Objects)
  - containsValue(Objects)

→ contains Value (Objects)

→ remove (Object)

keys	values
Earth	Planet
Venus	Planet
Andromeda	Galaxy
Milky Way	Galaxy
Sirius	Star
Europa	Moon
Sun	Star

Different Views

→ map has DS shared by other views

→ entrySet

→ entrySet
  → Entry<K, V>
  → Set because of unique elements
  <Earth, planet>, <Venus, planet>
  → operations:
    → contains, remove, size, clear
    → Object
    → own iterator

→ keySet
  → Set of keys
  → has same operations as entrySet
  → has own iterator
  Earth, Venus, Andromeda...

→ Values
  → AbstractCollection

→ Abstract Collection

→ has all collection methods

→ has own iterator

Planet, planet, galaxy, star...

Map class {

DS

constructor, etc

EntrySet class {

→ Constructor

methods

}

}

→ keySet & values will work based off entrySet

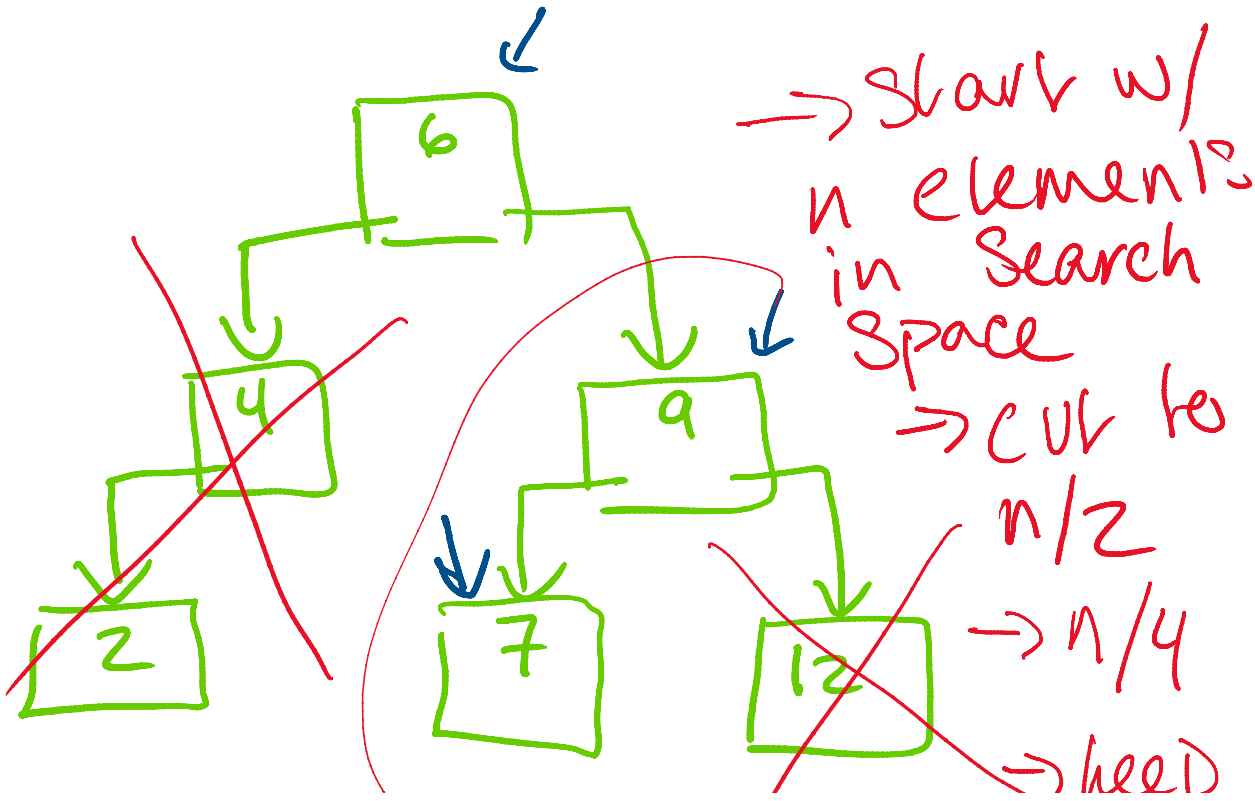
work based off entryset

Binary Search

2, 4, 6, 7, 9, 12

→ we want to search for "8"

→ basic: search one by one through list
→ O(n)



→ Search in BST:
O(log n)

→ Binary search uses same technique on arrays
→ Array must be sorted

9, 3, 7, 4, 2, 12

2, 3, 4, 7, 9, 12

high
low

→ low is inclusive, so move to mid+1 if going right
→ high is exclusive, so

→high is exclusive, so move to mid if going left

→can be implemented w/ recursion or a simple loop

→in lab: no recursion

*recursion is spatially expensive

→no advantage to using recursion for binary search

→doesn't significantly reduce code

→recursion can be difficult to read/debug