ProgTeam Week 5

Data Structures II: TreeSets and HashSets

Unordered Sets

aka Hash Set, Hash Table

- What if I only care if an element is present in a collection or not?
- O(N) time with a vector: check all elements
- Unordered Set: Assign each element a random hash function
  - Takes constant time to add/remove/find an element

Unordered Set

- Hash Function: a pseudorandom function H such that H(x) = H(y)
  - If two elements have same hash, we have to iterate through them to find a certain element
  - Goal: minimize collisions
- For integers, H(x) = x
- For strings, good hash is $(p^0 * s[0] + p^1 * s[1] + p^2 * s[2] + \ldots) \bmod m$ for prime ‘p’ and ‘m’
- Good news! Most libraries will hash strings for you, so we usually don’t need to worry about this
- Sometimes it’s useful to define a custom hash function

Unordered Set

- Most implementations use store a linked list for each unique hash “key”
  - If we want, we can store some other information that goes with the key

- This gives us an Unordered Map
  - Similar to an array, except we can have completely arbitrary indices

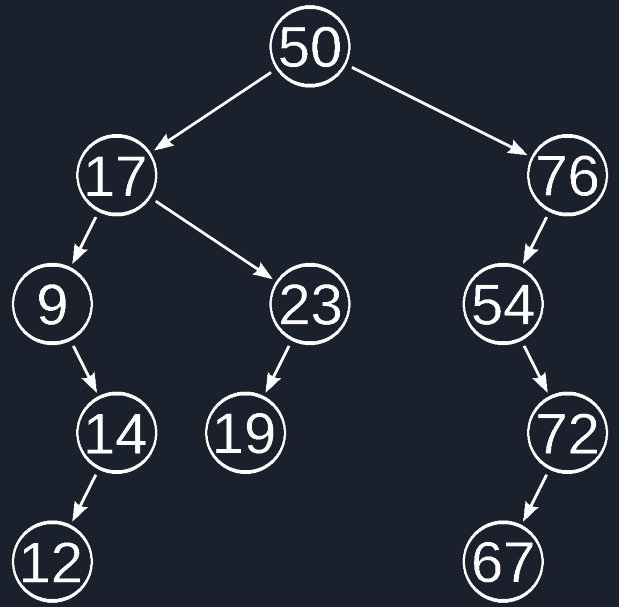
Ordered Set

aka Tree Set, Balanced Binary Search Tree (BBST), BST

- In some situations, the order of the elements can be useful
  - Maybe we want to know the element that's the next greatest compared to certain value
- An ordered set can answer questions like this in log(N) time

Ordered Set

- Store elements on a tree
- Time it takes to find element is height of tree
- Equal to log(N) for a balanced tree



Ordered Set

- Ordered Set is useful if you don’t have a hash function for your data type
- All these operations take Log(n) time:
  - Find
  - Insert
  - Remove
  - Upper Bound (which element comes after ‘x’?)
  - Lower Bound (which element comes before ‘x’?)

Multisets (or Unordered Multisets)

- By default in most languages, inserting an element already present does nothing
- We can use a Multiset to have several copies of the same element
  - Alternatively, we can tie an index to the original element and store a pair