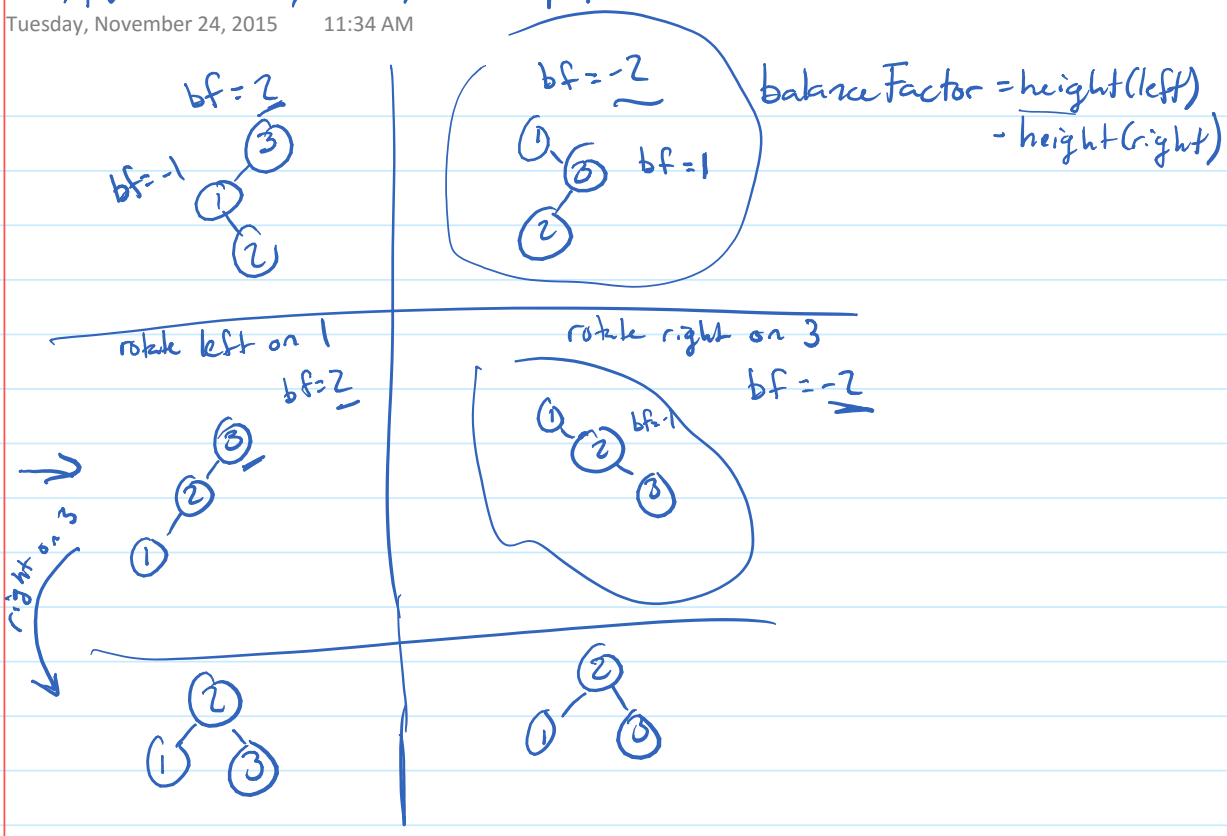
AVL Trees, Sets, Hashmaps, etc.

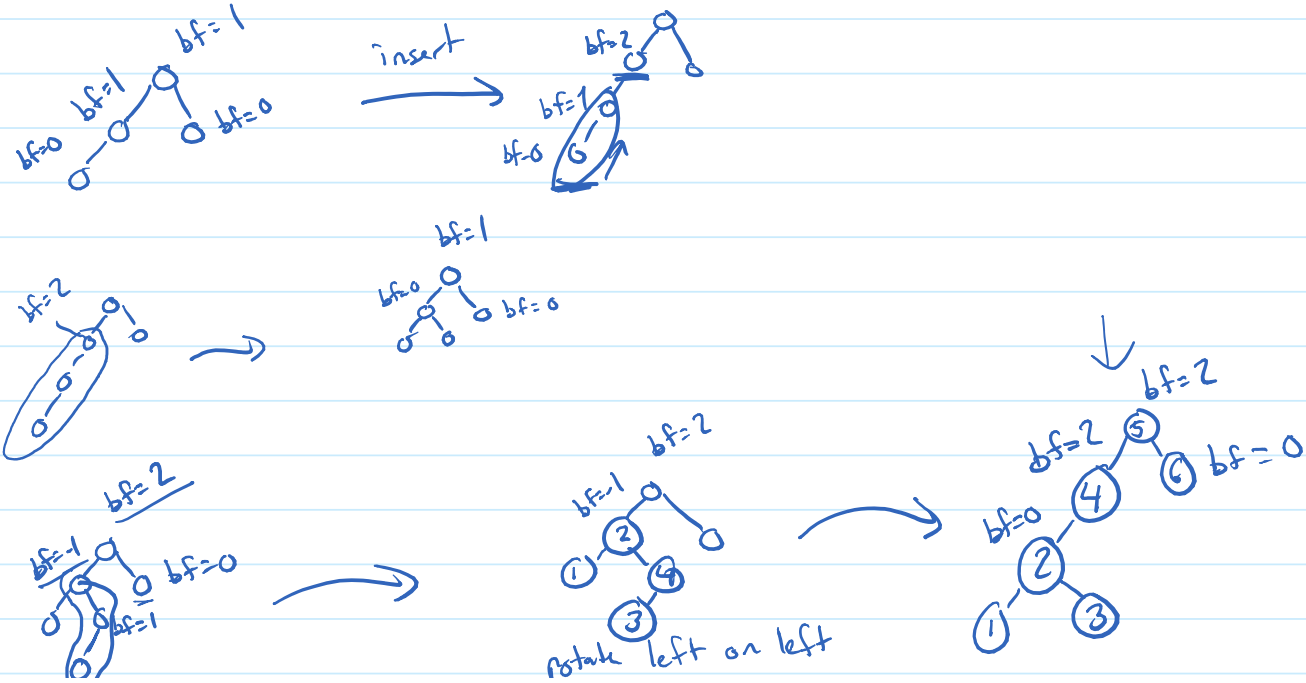
Tuesday, November 24, 2015 11:34 AM

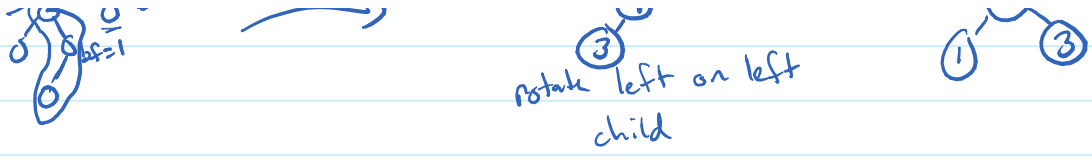


AVL trees

insert: do a standard BST insert

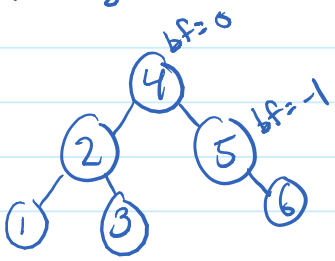
(Traverse back to the root, balancing as needed

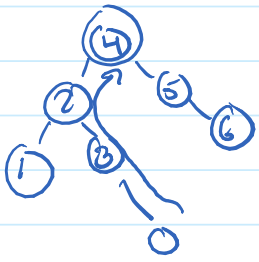




rotate right on 5

AVL Tree





Standard BST

Best way

```
Node {
    E element;
    Node left, right, parent;
    int balanceFactor;
}
```

$-2 \le bf \le 2$

3 bits

```
Node {
    E element;
    Node left, right, parent;
    int leftHeight, rightHeight;
}
```

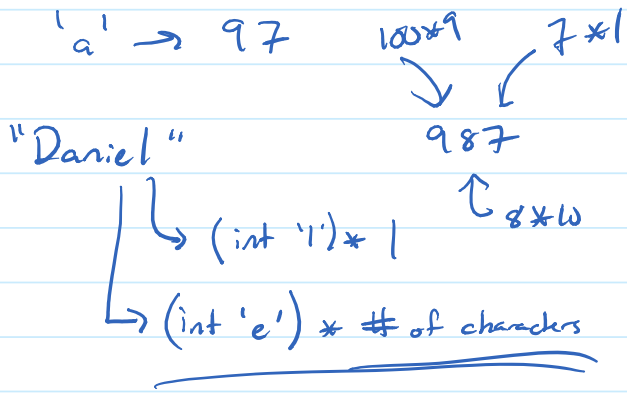
bigger than constant

Hash Maps, HashSets

Hash functions

Should take an object and produce some unique identifier for that object.

In Java "identifier" is an int



This generates a unique encoding, but it uses too much space

Reduce the size of the "range" of my hash function (i.e. multiple strings must map to same int)

md5
sha-1
sha-256

Hash Set

Naïve approach: Use a list
On insert, check for equality:
if no: insert
else: do nothing

Better approach:
- Use a list of hashcodes
- On insert, let hash be the hashcode of the obj:
  - Iterate through sublist at hash, checking for equality:
    - if no: insert
    - else: do nothing

Hash Map

E[] arrays hold some arbitrary type, but they're indexed w/ numbers

HashMap<K, V> (Dictionaries, assoc. lists)

grades["Dan"] = ∅ ⟼ grades.set("Dan", ∅);

list[1] = "Dan" ⟶ list.insert(1, "Dan")
wast

print grades["Dan"] ⟼ grades.get("Dan");

>0

import java.util.HashMap

HashMap<String, Integer> grades = new HashMap<String, Integer>();

public class HashMap<K, V> {

Most operations on HashMaps are constant

Objects
String toString()
boolean equals(Obj)
int hashCode()

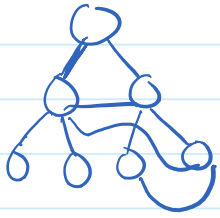
Graphs
Tree

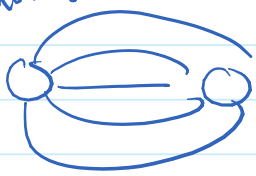
What is a graph?
a graph is a set of vertices, and a set of edges

Multigraph

of edges

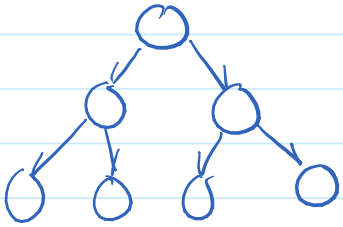
multigraph



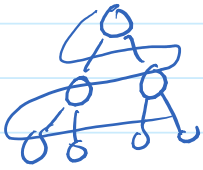


Undirected graphs

Directed graphs
"edges have a direction"



Undirected graphs are directed graphs where, given an edge in E from x to y, the corresponding edge from y to x is also in E



DFS & BFS extend to graphs