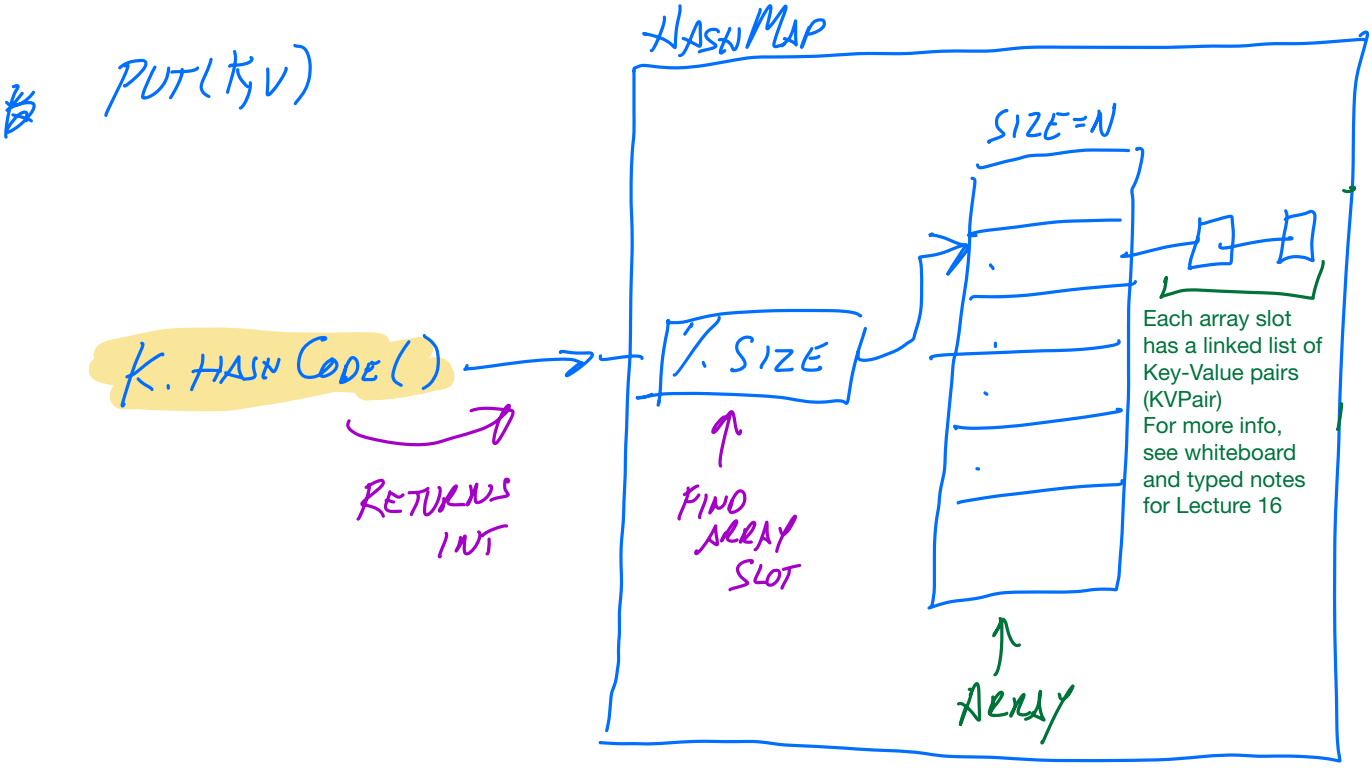
Hash maps: Programmer perspective

```
// Map people to office locations
HashMap<String, String> offices = new HashMap<String, String>();

// Associate this key with this value
offices.put("Lee", "CIT219");
offices.put("Sun", "CIT501");
offices.put("Helen", "B&H802");

offices.get("Sun");
offices.containsKey("Helen”);

// Iterating over a HashMap (one way)
for (String k : offices.keySet()) {

}
```

Programmer perspective
- Each key can only map to one value
- For all operations, Java calls hashCode() on the key to get an integer value
  - All keys with the same code map to the same value
- Java already has a hashCode for built-in types—if you’re making your own class, you should make your own hashCode (like equals())

Hashmaps: Implementation perspective



What would the hash map look like after the following operations?

```
offices.put("Lee", "CIT219");
offices.put("Sun", "CIT501");
offices.put("Helen", "B&H802");
```

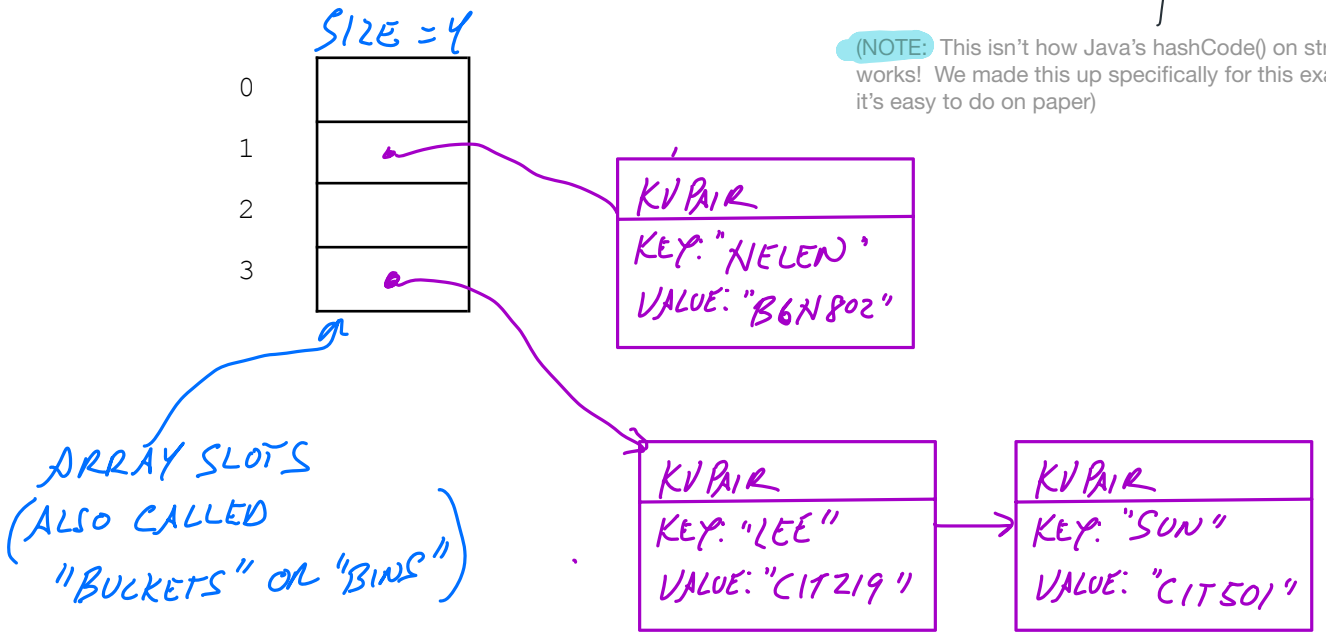
PENSON

Assume: array size 4, hashCode() returns the length of the key string

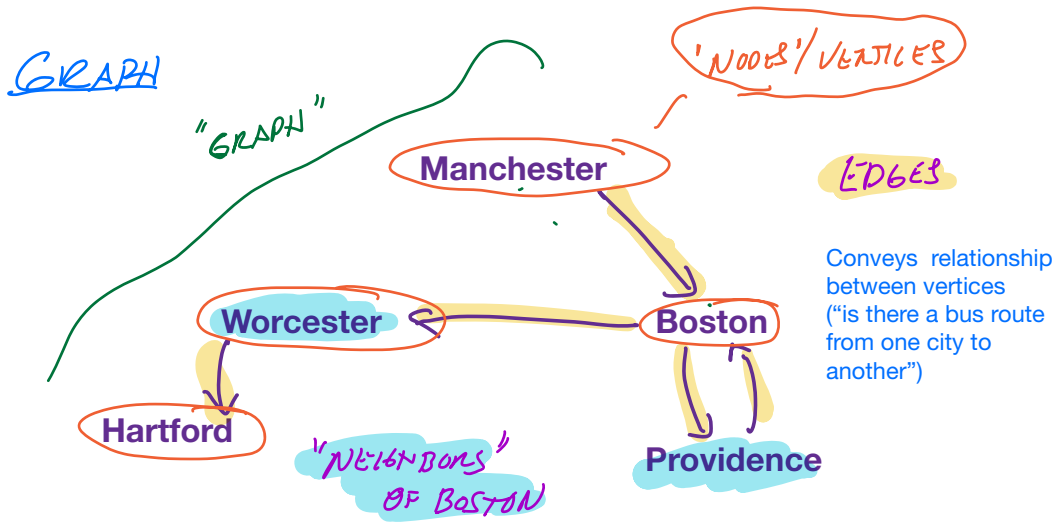
"ABCD"

Key	key.hashCode()	Array slot (% 4 in this example)
LEE	3	3
SUN	3	3
HELEN	5	1

(NOTE: This isn't how Java's hashCode() on strings works! We made this up specifically for this example so it's easy to do on paper)



(These KVPair objects are part of a LinkedList. To keep the diagram clean, we haven't drawn the LinkedList objects here.)



Conveys relationship between vertices ("is there a bus route from one city to another")

How is this different from a tree?
- No inherent hierarchy like trees (no "root")
- Can have "cycles"

What kinds of classes, datatypes etc would you use to represent this?

Some questions/options
- For each each node, have a List of neighbors
- Should each node be a class? Or just, eg, a string?
- List of String -> String pairs to represent each edge

```java
public class CityVertex {
    LinkedList<CityVertex> toCities;
    String name;

    public CityVertex(String nm) {
        this.name = nm;
        this.toCities = new LinkedList<CityVertex>();
    }

    public void addEdge(CityVertex toVertex) {
        this.toCities.add(toVertex);
    }

    public String toString() {
        String retstring = "City " + this.name + " goes to { ";
        for (CityVertex toCity : this.toCities) {
            retstring += toCity.name + " ";
        }
        retstring += "}";
        return retstring;
    }
```

A NODE

LIST OF NEIGHBORS

ADD A NEIGHBOR

```java
public class TestCityVertex {
   public static void main(String [] args) {
       CityVertex man = new CityVertex("Manchester");
       CityVertex bos = new CityVertex("Boston");
       CityVertex pvd = new CityVertex("Providence");
       CityVertex wos = new CityVertex("Worcester");
       CityVertex har = new CityVertex("Hartford");


        man.addEdge(bos);
        bos.addEdge(pvd);
        bos.addEdge(wos);
        pvd.addEdge(bos);
        wos.addEdge(har);

        System.out.println(bos);









   }
}
```

ADDING EDGES TO CREATE OUR GRAPH