**2022FA Data Structures (CS-215-ON)**

**Greg Gutierrez**

**Part 1:**

1. **Explain how a hash function is used.**

A hash function is a method that inspects the contents of an object and returns an integer value such that things with identical contents will return the same hash code Objects with different contents will, with high probability, return other hash codes. Another way to state these principles is that an ideal hash function is a deterministic function whose value, for random input, is uniformly distributed over the range of (finite) integers. One hard and fast rule of designing a hash function is that every bit of the object’s contents must influence the hash code. If some contents are ignored, it increases the probability that objects with different ranges will yield the same hash code.

1. **How might a string hash function be written?**

A hash function for String objects might be written as

int hashString(String s) {

int hash = 0;

for (int i = 0; i < s.length(); i++) {

hash += s.charAt(i);

}

return hash;

}

The problem with additive hash codes is that they do a poor job of preventing collisions: in particular, in an additive hash code, the order of the primitive values is not significant. For this reason, the following collisions occur:

hashString("ab") == hashString("ba")

hashString("dad") == hashString("add")

Another problem is that there are generally many pairs of integers summing to the same value: so we also would have collisions like

hashString("ad") == hashString("bc")

A better approach is to return to the original idea of viewing the contents of the object as a string of "digits":

N = C

n-1 ⋅ s1 + Cn-2 ⋅ s2 + Cn-3 ⋅ s3 + ... + C1 ⋅ sn-1 + C0 ⋅ sn

Because we are using finite integer arithmetic, we aren't guaranteed to get a unique hash code for every distinct object, but that's OK if the distribution of hash codes is reasonably uniform. This approach will be somewhat expensive to compute because of the need to raise the constant C to various powers. However, that is not necessary because we can rewrite the formula for any number of "digits" as follows:

C3 ⋅ s0 + C2 ⋅ s1 + C1 ⋅ s2 + C0 ⋅ s3 = C ⋅ (C ⋅ (C ⋅ (s0) + s1) + s2) + s3

So, we need one addition and one multiplication per "digit.” So, a better hash function for Strings would be:

int hashString(String s) {

int hash = 0;

for (int i = 0; i < s.length(); i++) {

hash \*= C;

hash += s.charAt(i);

}

return s;

}

1. **Explain why we might use a hash function rather than search for a key.**

A hash function takes a key as an input associated with a datum or record and is used to identify it to the data storage and retrieval application. The keys may be fixed length, like an integer, or variable length, like a name. In some cases, the key is the datum itself. The output is a hash code to index a hash table holding the data, records, or pointers. 1) it should be very fast to compute; 2) it should minimize duplication of output values (collisions). Finally, the complexity is O(1)

1. **What hash function does the Java Util HashMap use for**[**hashing**](https://moodleilp.bellarmine.edu/mod/url/view.php?id=778792)**strings?**

To access a value, one must know its key. HashMap is known as HashMap because it uses a technique called Hashing.[Hashing](https://www.geeksforgeeks.org/hashing-set-1-introduction/) is a technique of converting a large String to a small String representing the exact String. A shorter value helps in indexing and faster searches. [HashSet](https://www.geeksforgeeks.org/hashset-in-java/) also uses HashMap internally.

A few essential features of HashMap are:

* HashMap is a part of the java. util package.
* HashMap extends an abstract class, AbstractMap, which also provides an incomplete implementation of the Map interface.
* It also implements a [Cloneable](https://docs.oracle.com/javase/7/docs/api/java/lang/Cloneable.html) and [Serializable](https://docs.oracle.com/javase/7/docs/api/java/io/Serializable.html) interface. K and V in the above definition represent Key and Value, respectively.
* HashMap doesn’t allow duplicate keys but allows the same values. That means A single key can’t contain more than one value, but more than one key can have a single value.
* HashMap allows null keys also but only once and multiple null values.
* This class makes no guarantees as to the order of the map; in particular, it does not guarantee that the order will remain constant over time. It is roughly like HashTable but is unsynchronized.

References:

<https://www.geeksforgeeks.org/>

<https://www.ionos.com/digitalguide/server/security/hash-function/>