Hash Tables

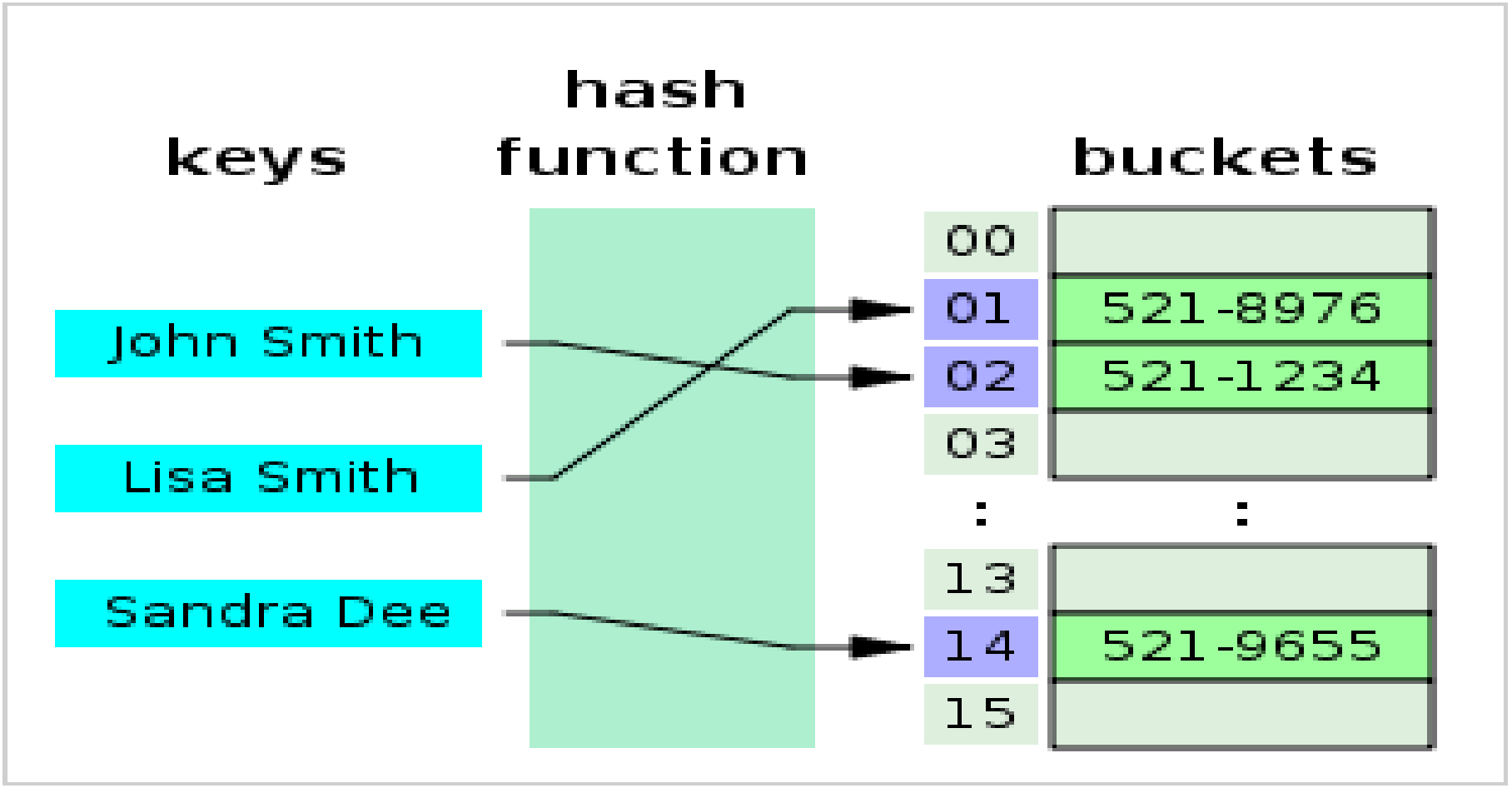
Information resource for some of the following material
http://en.wikipedia.org/wiki/Hash_table

Hash Table

A hash table (or hash map) is a data structure that maps keys (identifiers) into a certain location (bucket)

A hash function changes the key into an index value (or hash value)

Graphical View of Hash Table



Collisions

Ideally, each key maps to an open bucket in the hash table (perfect hash)

In most cases, a perfect hash is not attainable, so collisions occur where two different keys map to the same hash location

We will need to define collision handling techniques

Hash Functions

Hash function – needs to compute the symbol's bucket address from the symbol itself

i.e. some function h(K) maps the domain of keys K into a range of addresses 0, 1, 2, … M-1

The Problem

- Finding a suitable function h
- Determining a suitable M
- Handling collisions

Hash Methods

Many hash methods exist. Let's start with two easy methods:

Mid Square

1)Square the key value
2)Take a certain number of bits from the middle of the squared value to form a bucket address

Idea – all characters making up the key are used in this process so that similar keys will have fewer collisions

Midsquare Hashing

Problem: Let's assume that the key value is simply the sum of the ASCII values squared. If the key value is 16-bits and we take the middle 8-bits:

a) How big is the hash table?
b) What is the range of bucket addresses?
c) Where does the key AB map to in the hash table?

Division Hashing

Divide the key by some number N where N is a prime number corresponding to the number of buckets in the hash table.

a) How big is the hash table?
b) What is the range of bucket addresses?
c) Where does the key AB map to in the hash table?

Collision Handling

Open Addressing

Suppose we have a key K such that h(K) maps to the same location as key K' which is distinct from K.

To resolve this conflict in open addressing, find another unoccupied space for key K'

Open Addressing

If b0=h(K), the let b0, b1, b2, ..., bM-1 be a probe sequence where M = tablesize

Hash Table Search by Open Addressing is as follows:

Let T be some table with M entries which looks like: T[0],T[1],T[2],...,T[M-1]. We will assume without loss of generality that all keys inserted have positive values and empty entries are signified by a value of 0. We will search for the key K. The following algorithm will perform this search

Search Algorithm

```
i := h(K)
j := i
while((T[i].key <> K) and
      (T[i].key <> 0)) do
  begin
    i := i - p(K)
    if(i < 0) then i := i + M
    if(j = i) then tablefull(i)
  end
return i
```

The question is: How do we choose p(K)?

Linear Probing

With linear probing we choose p(K) = 1 which implies the following: h(K), h(K)-1, h(K)-2,...,0,M-1,M-2,...,h(K).

Problem: Using the hash function h(Kn) = n mod 11 show what the hash table would look like after inserting the following keys: M13,G7,Q17,Y25,R18.

Just use the number following the letter as the key value.

What happens if you add Z26 and then F6?