Hashing

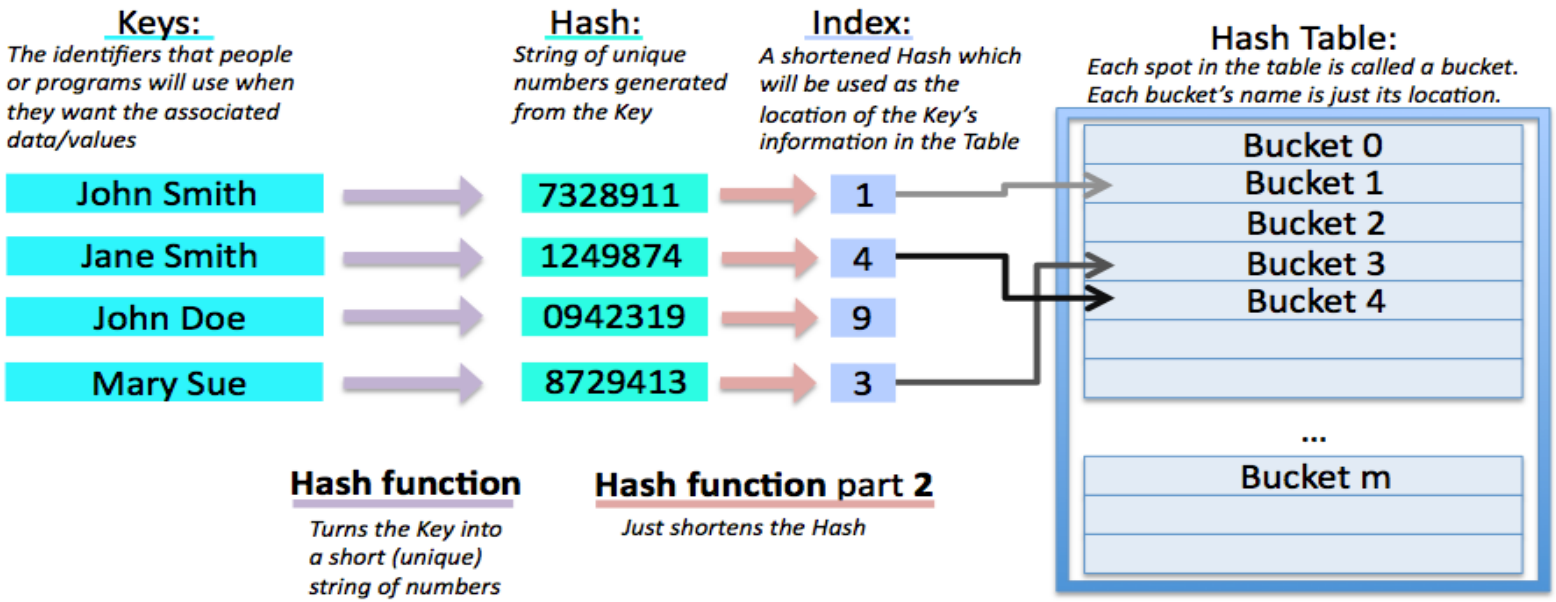
Jaafar Alzubaidi
Ioannis Karagiannis
Shaoling Zhu

Hashing

- Hash table
- Hash functions
- Collisions
- Simple example

Hash Table

- Data structure used to implement associative array
- Maps keys to values using a hash function



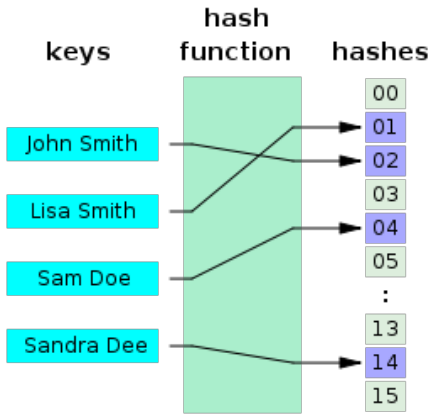
Hash Table

- Faster than other data structures when searching
- Increases speed while searching, comparing and inserting data
- Collisions (later)

10	21	22	33	54	5	2376	8887	28	9

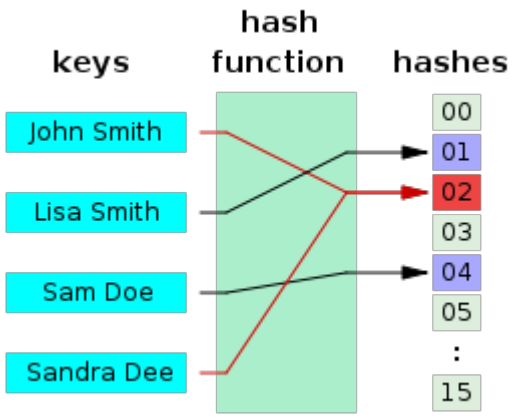
Hash Functions

- Any function to map digital data of arbitrary size to digital data of fixed size
- Accelerate table search or database lookup by detecting duplicated records
- Choice of hash function depends on nature of input data
- Some hash function algorithms:
  - –Perfect hashing
  - –Hashing uniformly distributed data
  - –Hashing variable-length data



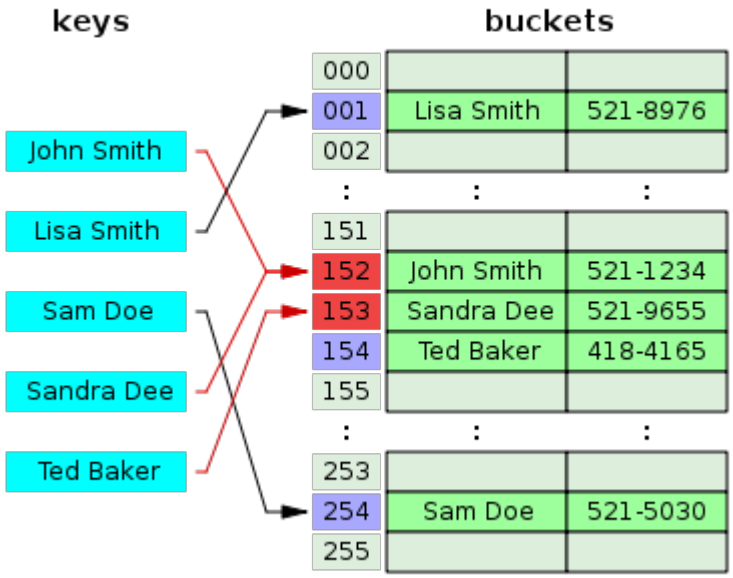
Collisions

- Collisions are unavoidable
- Strategies to handle collisions
  - –Open addressing
    - •Linear probing
    - •Quadratic probing
  - –Separate chaining



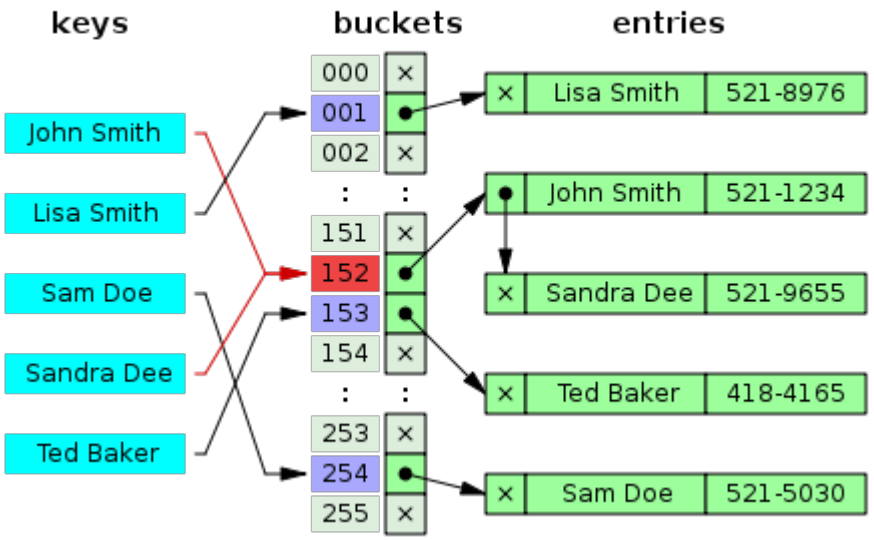
Collisions

- Open addressing
  - Linear probing



Collisions

- Separate chaining



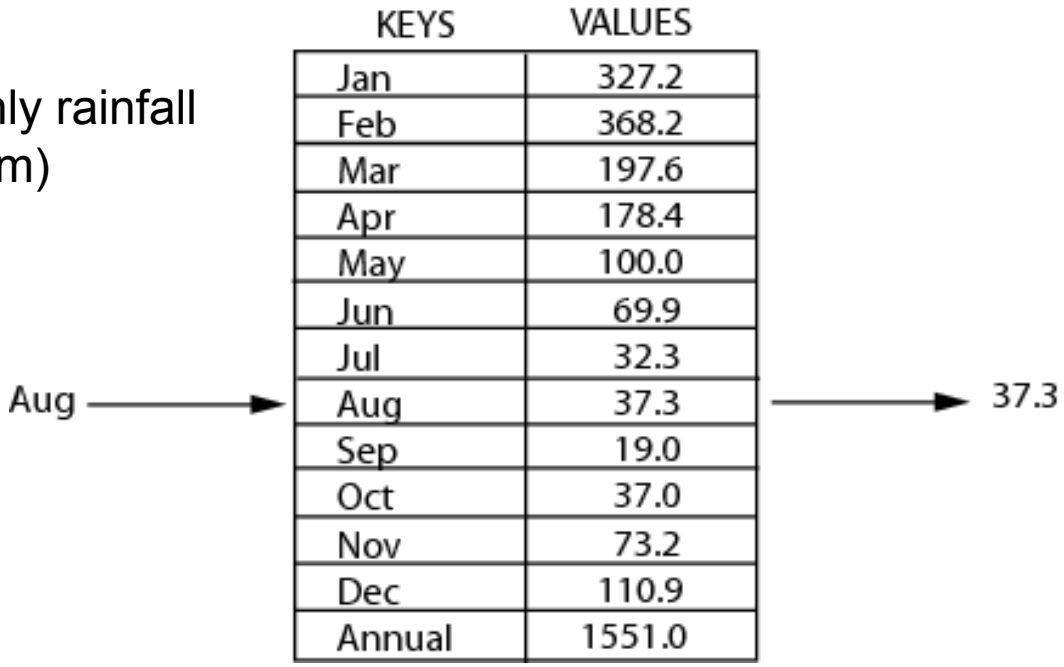
Code Example of Hash function

- Redirecting to Matlab...

Map Containers

- Fast key lookup data structure
- Map values to unique keys
- Retrieve values with a corresponding key

Mean monthly rainfall statistics (mm)



Properties of the Map Class

mapObj = containers.Map(keySet, valueSet)

mapObj.Count

mapObj.KeyType

mapObj.ValueType

Property	**Description**	**Default**
Count	Uint64	0
KeyType	double, single, char	char
ValueType	any	any

Methods of the Map Class

Method	Description
isKey	Check if Map contains specified key
keys	Names of all keys in Map
length	Length of Map
remove	Remove key and its value from Map
size	Dimensions of Map
values	Values contained in Map

Code Example of Map Class

- Redirecting to Matlab...

Locality-Sensitive Hashing (LSH)

- What is LSH all about?
  - A method of projecting probabilistically high dimensional-data into a fewer dimensional-space

- In what sense “Hashing”?
  - Hash functions to maximize the probability of collision of similar data (the opposite of a regular hash function’s objective)

LSH (general principles)

1. Take random projection of data (low-dimensional binary space→ ***Hamming*** space)

2. Associate each data point to a b-bit vector (hash key)

3. $Pr[h(x_i)=h(x_j)] = sim(x_i,x_j)$

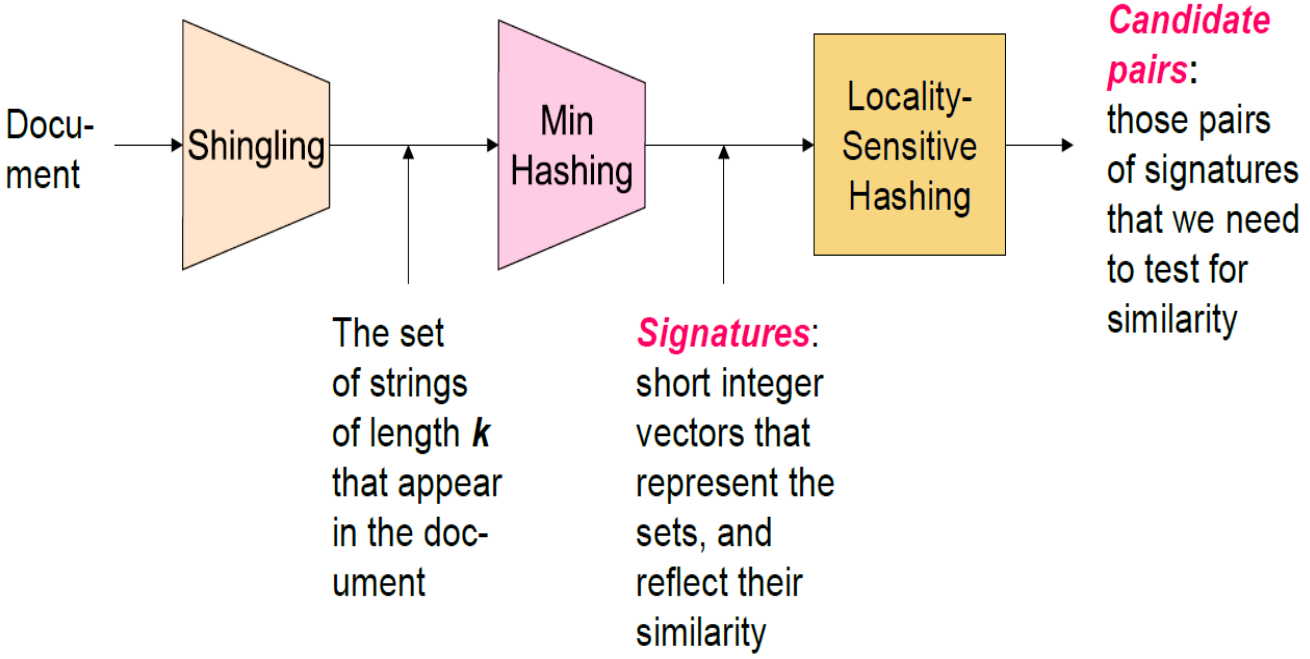
Basic idea is that the number of buckets will be significantly less than the universe of possible input arguments.

LSH Applications

- Nearest neighbor search
- Audio similarity identification
- Image similarity identification
- Near-duplicate detection
- Hierarchical clustering
- Audio fingerprint

Find pairs of similar docs

- Hash similar documents to the same buckets
- Only compare candidate documents (hashed to the same bucket)
- Only *O(N)* comparisons instead of *O(N^2)*



Shingling and Min-Hashing

- Take consecutive words, group them together as a single object. A ***k-shingle*** is a consecutive set of k words. e.g.:

  D_1: I am Jack

  D_2: I am a gambler

  (k=1)-shingle of $D_1 \cup D_2$: { [I], [am], [Jack], [a], [gambler] }

  (k=2)-shingle of $D_1 \cup D_2$: { [I am], [am Jack], [am a], [a gambler] }

  JS(k=2) = $|D_1 \cap D_2| / |D_1 \cup D_2|$ = ¼ = 0.25 (***Jaccard*** similarity)

  JS(k=1) = $|D_1 \cap D_2| / |D_1 \cup D_2|$ = ⅖ = 0.40

- ***Min-Hashing*** is a technique to estimate the similarity of two sets (*Andrei Broder* 1997)

Thanks for your attention!