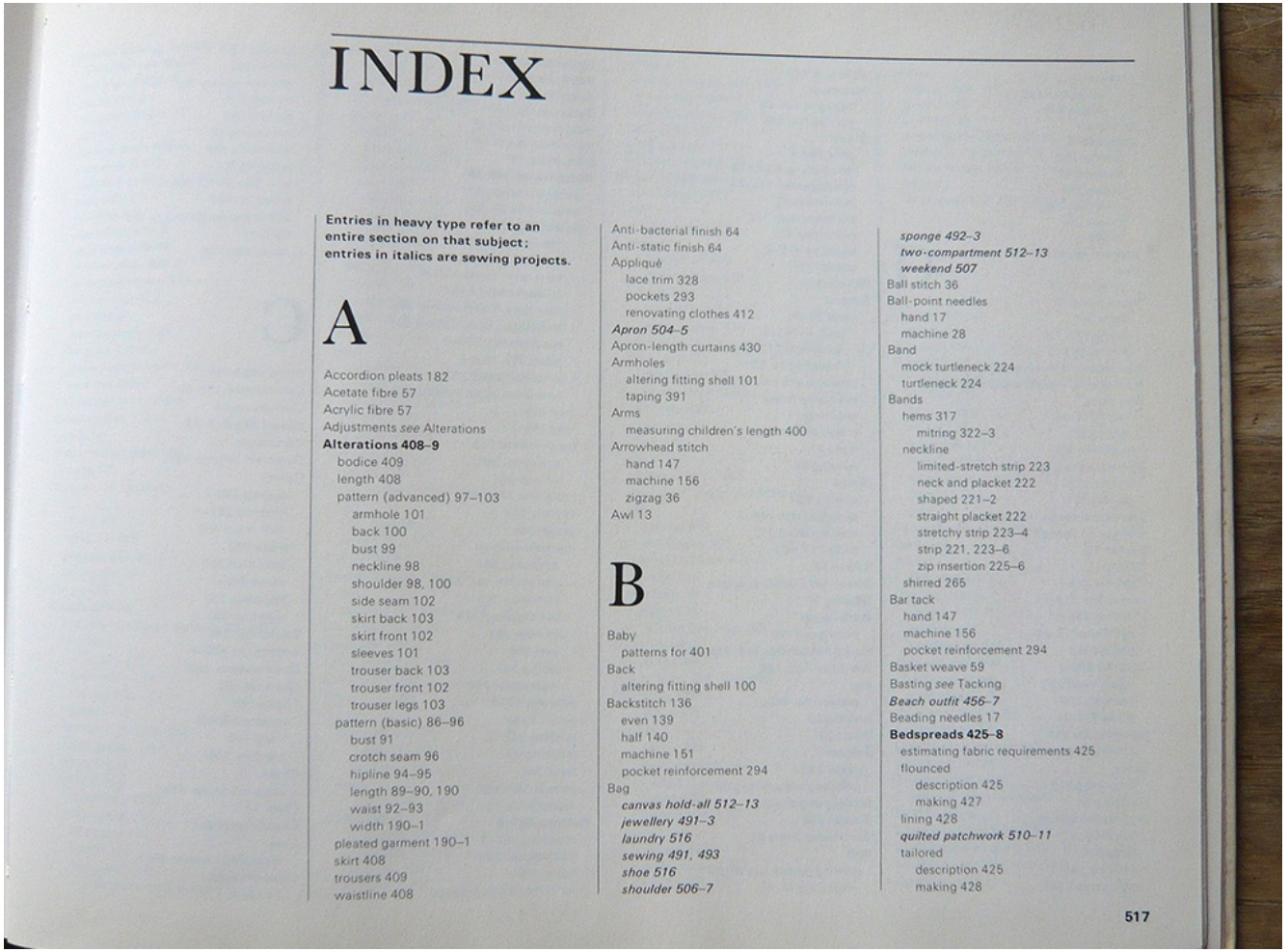
Indexing

Class Algorithmic Methods of Data Mining
Program M. Sc. Data Science
University Sapienza University of Rome
Semester Fall 2015
Lecturer Carlos Castillo http://chato.cl/

Sources:

- Slides by Silberschatz et al. 2006
- Fundamental data structures Wikibook

Index



INDEX

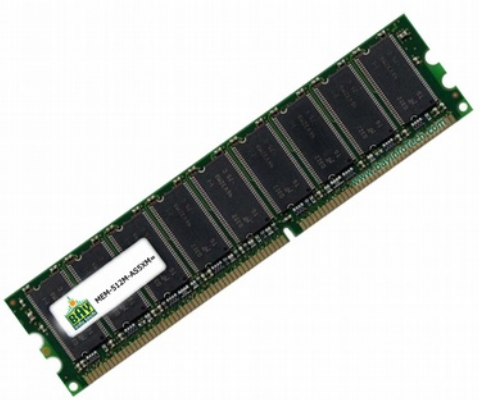
Entries in heavy type refer to an entire section on that subject; entries in italics are sewing projects.

A

B

What is an index?

Why an index?



Sequential files for storing items (e.g. documents)

(

record-code (20 bytes, e.g. “REU20151014115203001”)

record-length (4 bytes)

contents (*record-length* bytes)

)+

- Problem: accessing a specific item requires to scan the entire file

Index

- Physical storage of data cannot be assumed to follow the order in which we want to access records
- Indices speed up data access
- Pairs of <search key, pointer>
- Ordered indexes (such as the index in a book)
  - Entries are sorted
- Hashed indexes
  - Entries are on a hash table

Example dense index

Record code	Record number
REU2015101411520300 1	1
CNN2015101411549100 1	2
CNN2015101411549100 2	3
BBC2015101523170400 1	4
...	...

Record code	File number, Record number
REU2015101411520300 1	1, 1
CNN2015101411549100 1	1, 2
CNN2015101411549100 2	1, 3
BBC2015101523170400 1	2, 1
...	...

Is this index helpful? Yes, no, why?

Example sorted dense index

Record code	File number, Record number
BBC20151015231704001	2, 1
CNN20151014115491001	1, 2
CNN20151014115491002	1, 3
REU20151014115203001	1, 1
...	...

What can we do when the index becomes too big to fit in main memory?

Example sorted 2-level dense index

Record code	Index number
BBC2015101523170400 1	2
CNN2015101411549100 1	1
CNN2015101411549100 2	1
REU2015101411520300 1	1
...	...

Record code (index 1)	Record number
CNN2015101411549100 1	2
CNN2015101411549100 2	3
REU2015101411520300 1	1
...	...

Record code (index 2)	Record number
BBC2015101523170400 1	1
...	...

What can we do when the first-level index becomes too big to fit in main memory?

Example sorted 2-level sparse index

Record code	Index number
BBC20151015231704001	2
CNN20151014115491001	1
REU20151014115203001	1
...	...

Record code (index 1)	Record number
CNN20151014115491001	2
CNN20151014115491002	3
REU20151014115203001	1
...	...

We know CNN20151014115491002 goes after CNN20151014115491001 and before REU20151014115203001, we don't include it in the first-level index if its index number is the same as before. Many other tricks are possible.

How many operations to find a record in a (first-level, dense) index?

How do you find a word?

You go to the middle, look for the word there then go to the middle of the corresponding part … until finding the word.

If there *n* words, you have multiplied *n* by ½ until it became 1.

How many operations?

How many operations to find a record in a (first-level, dense) index?

How do you find a word?

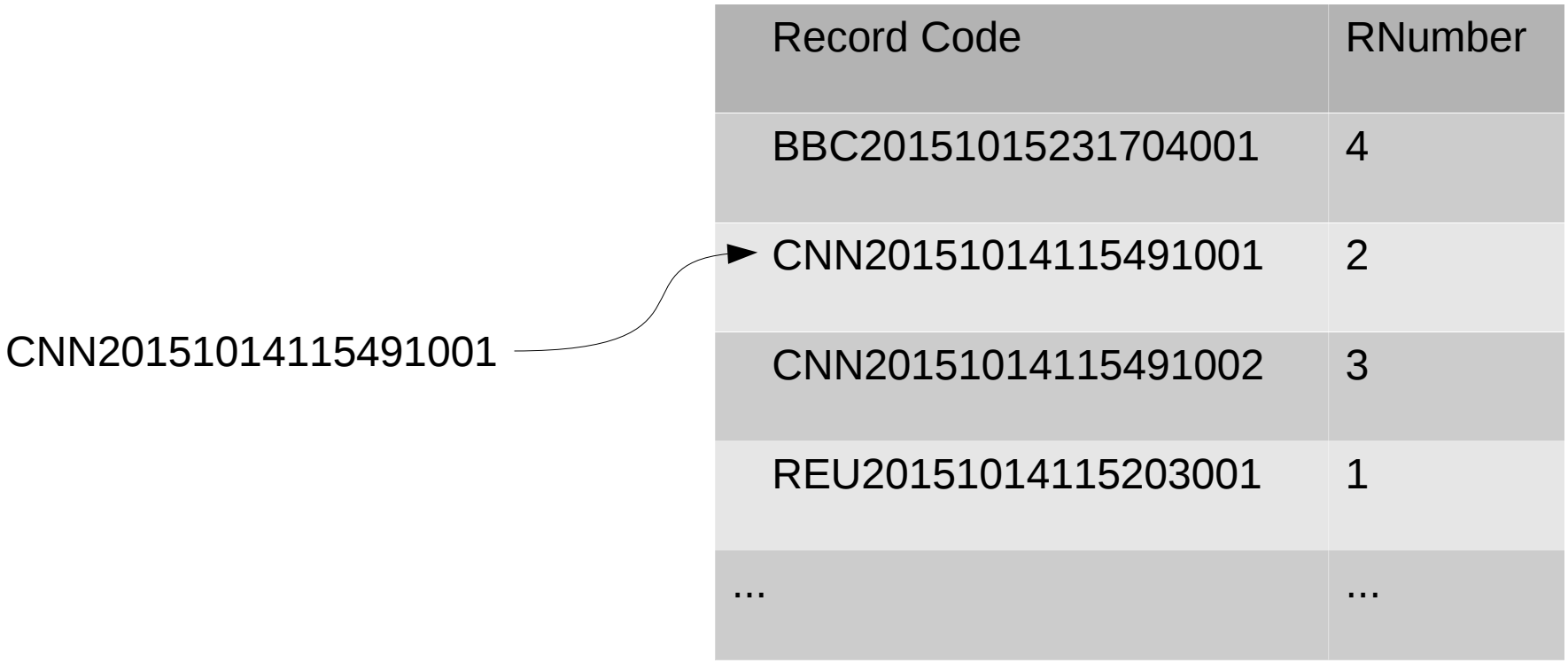
You go to the middle, look for the word there then go to the middle of the corresponding part … until finding the word.

If there *n* words, you have multiplied *n* by ½ until it became 1.

This is about $log_2(n)$ operations.

Hashing

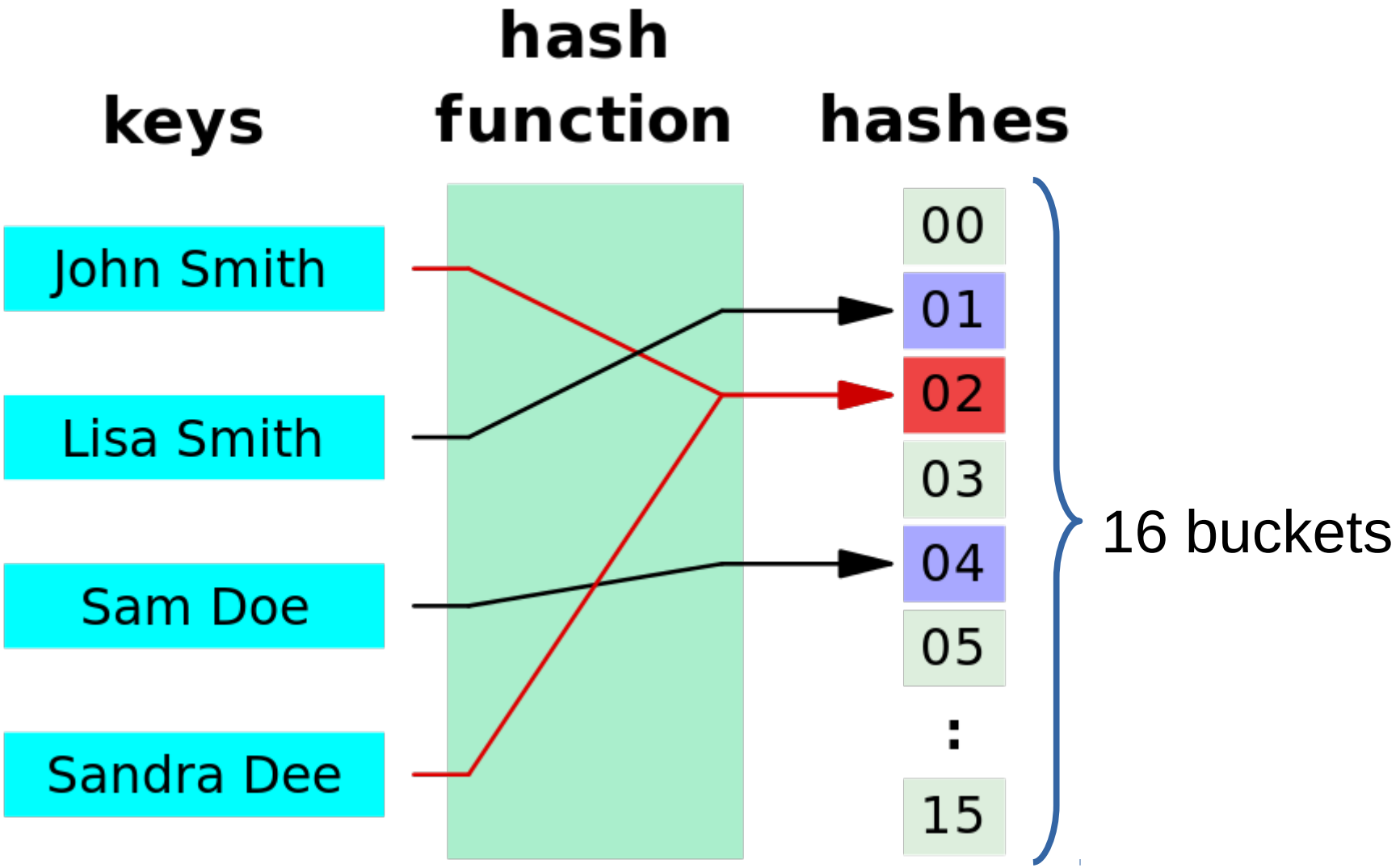
Reducing search time from $\log_2(n)$ to 1 operation



$|U|$ keys

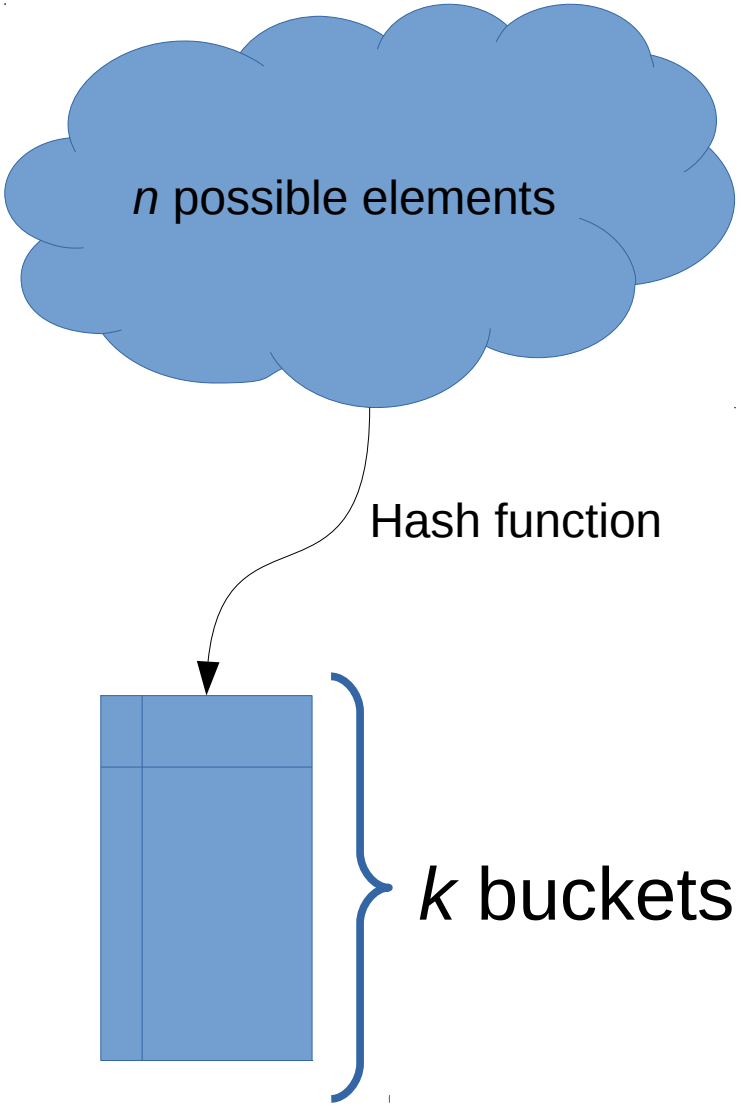
k buckets

Example hash table for names



Hashing

- Invented in 1953
- Map a large space of keys to a much smaller space of buckets
- E.g. keys = all the possible names of people
- E.g. values = numbers from 1 to 1,000
- **All buckets should be close to equiprobable**



Example hash function

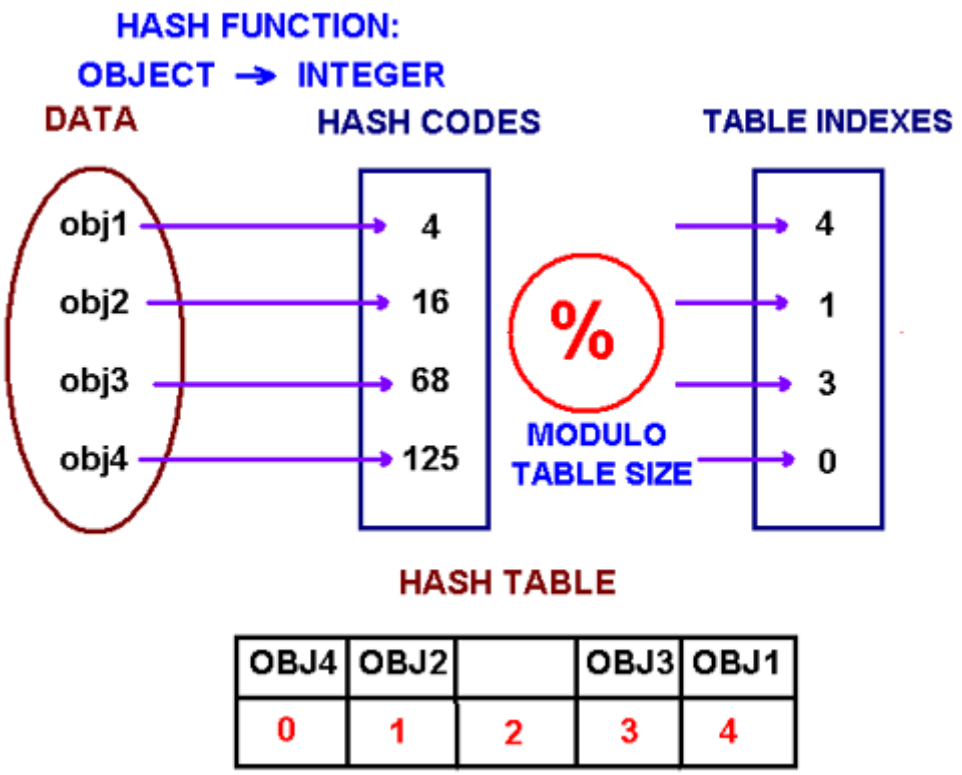
Problem: bucket space is too big

Example hash function ($k=26^6$)

h(“Maria Rossi”) = “RSSMRA”

Problem: still too big

Typical trick: design a good hash coding, then apply modulo table size



http://www.cs.cmu.edu/~adamchik/15-121/lectures/Hashing/hashing.html

Example hash function
(k=1,000,000, to fit easily in main memory)

h(“Maria Rossi”) = numval(“RSSMRA”) mod 1000000

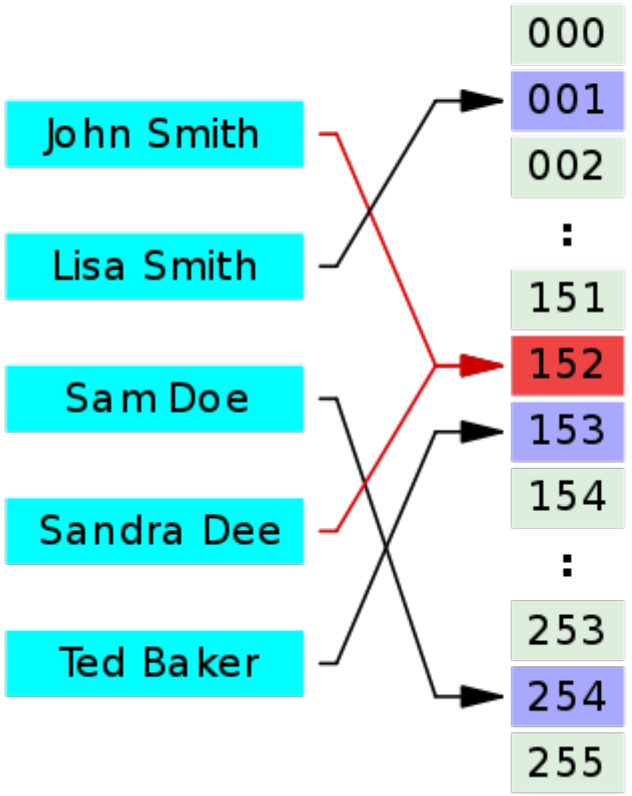
What is the problem with this hash function?

A simple hash function for strings

```
function hash(s, k) {
    int val = 0;
    for( int i=0; i<length(s); i++ ) {
        val = 13 * val + s[i];
    }
    return val % k;
}
```

Collisions

Keys (n) Buckets (k)

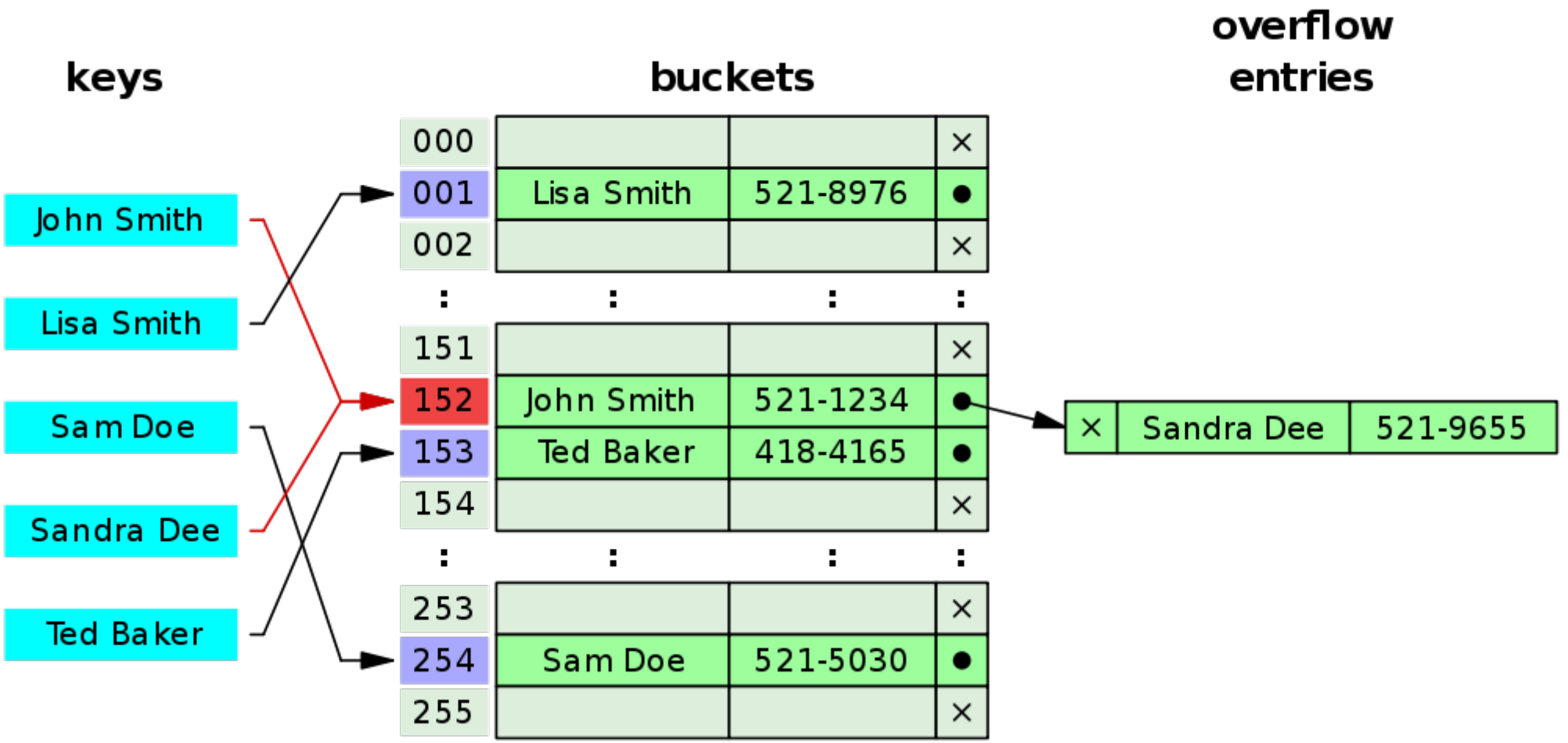


What shall we do about collisions?

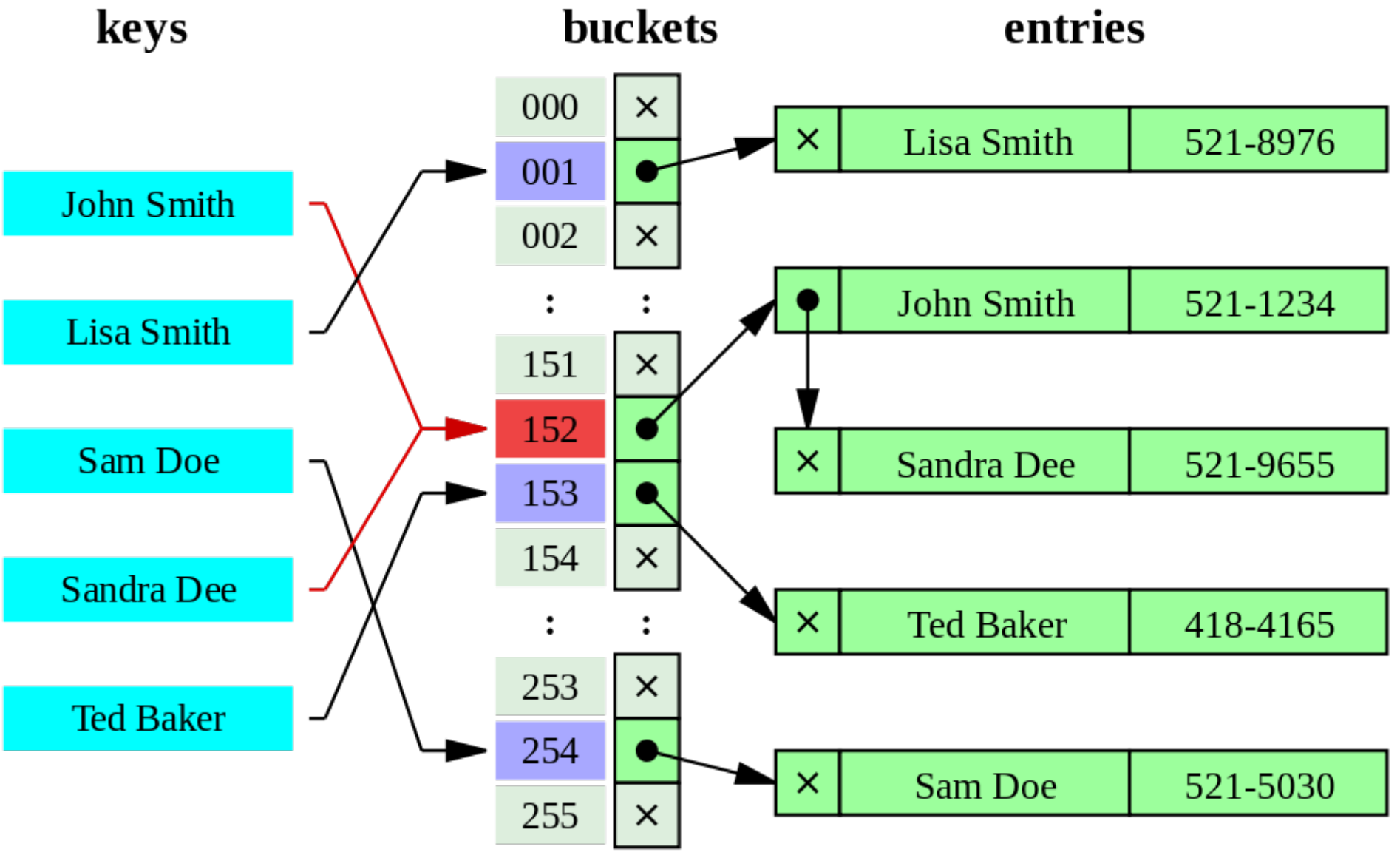
If $n < k$, can collisions be completely avoided?

See https://en.wikipedia.org/wiki/Perfect_hash_function

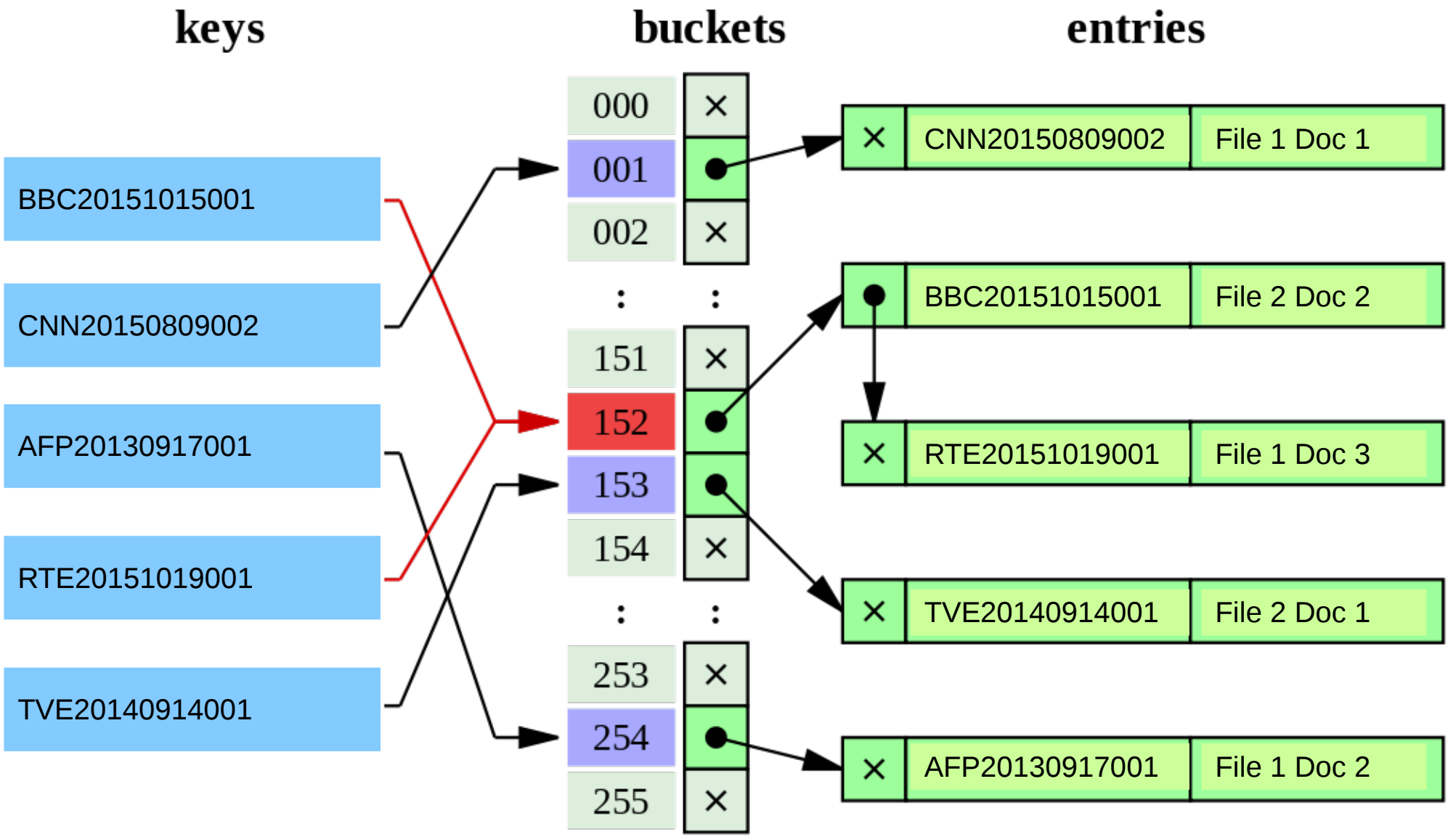
What to do with collisions? Solution 1: overflow entries



What to do with collisions? Solution 2: separate chaining



Back to our document system



Performance of hashing table

- Search
- Insert
- Delete

How many operations are required for any of these?

What factors make a hash table faster/slower?

Exercise

```
function hash(s, k) {
    int val = 0;
    for( int i=0; i<length(s); i++ ) {
        val = 13 * val + s[i];
    }
    return val % k;
}
```

Assume the value of s[i] = 1, 2, 3, 4 if s[i] is “a”, “b”, “c”, “d”

Create a hash table of size 4 for elements “a”, “ca”, “ac”, “ba”

http://chato.cl/2015/data_analysis/exercise-answers/indexing_exercise_01_answer.txt