

Topic 22

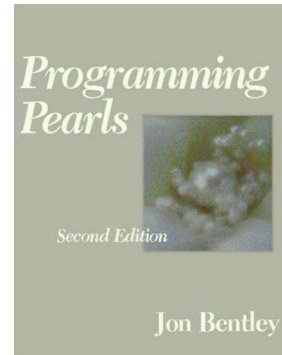
Hash Tables

"**hash collision** n. [from the techspeak] (var. `hash clash') When used of people, signifies a confusion in associative memory or imagination, especially a persistent one (see [thinko](#)).

True story: One of us was once on the phone with a friend about to move out to Berkeley. When asked what he expected Berkeley to be like, the friend replied: 'Well, I have this mental picture of naked people throwing Molotov cocktails, but I think that's just a collision in my hash tables.'

-The Hacker's Dictionary

Programming Pearls by Jon Bentley



- ▶ Jon was *senior programmer* on a large programming project.
- ▶ Senior programmer spend a lot of time helping junior programmers.
- ▶ Junior programmer to Jon: "I need help writing a sorting algorithm."

A Problem

- ▶ From *Programming Pearls* (Jon in Italics)

Why do you want to write your own sort at all? Why not use a sort provided by your system?

I need the sort in the middle of a large system, and for obscure technical reasons, I can't use the system file-sorting program.

What exactly are you sorting? How many records are in the file?

What is the format of each record?

The file contains at most ten million records; each record is a seven-digit integer.

Wait a minute. If the file is that small, why bother going to disk at all? Why not just sort it in main memory?

Although the machine has **many megabytes of main memory**, this function is part of a big system. I expect that I'll have only about a megabyte free at that point.

Is there anything else you can tell me about the records?

Each one is a seven-digit positive integer with no other associated data, and no integer can appear more than once.

System Sort

```
pisces% cat simple.txt
zoo
apple
bee
Apple
Zoo
Yacht
Soccer
2410 Speedway
Dorr
!!
pisces%
```



```
pisces% sort simple.txt
!!
2410 Speedway
apple
Apple
bee
Dorr
Soccer
Yacht
zoo
Zoo
```

Starting Other Programs

getRuntime

```
public static Runtime getRuntime()
```

Returns the runtime object associated with the current Java application. Most of the methods of class `Runtime` are instance methods and must be invoked with respect to the current runtime object.

Returns:

the `Runtime` object associated with the current Java application.

Starting Other Programs

exec

```
public Process exec(String command) throws IOException
```

Executes the specified string command in a separate process.

This is a convenience method. An invocation of the form `exec(command)` behaves in exactly the same way as the invocation `exec(command, null, null)`.

Parameters:

`command` - a specified system command.

Returns:

A new `Process` object for managing the subprocess

Clicker 1 and 2

▸ When did this conversation take place?

- A. circa 1965
- B. circa 1975
- C. circa 1985
- D. circa 1995
- E. circa 2005

▸ What were they sorting?

- A. SSNs. B. Random values C. Street Addresses
- D. Personal Incomes E. Phone Numbers

A Solution

```
/* phase 1: initialize set to empty */  
for i = [0, n)  
    bit[i] = 0
```

```
/* phase 2: insert present elements into the set */  
for each num_in_file in the input file  
    bit[num_in_file] = 1
```

```
/* phase 3: write sorted output */  
for i = [0, n)  
    if bit[i] == 1 write i on the output file
```

Some Structures so Far

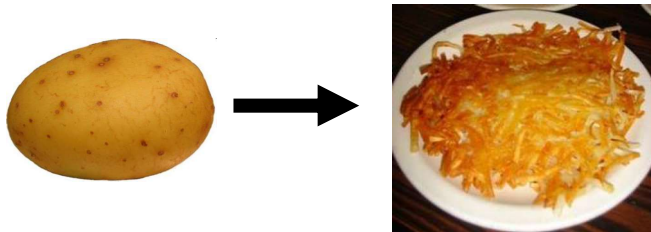
- ▶ ArrayLists
 - $O(1)$ access
 - $O(N)$ insertion (average case), better at end
 - $O(N)$ deletion (average case)
- ▶ LinkedLists
 - $O(N)$ access
 - $O(N)$ insertion (average case), better at front and back
 - $O(N)$ deletion (average case), better at front and back
- ▶ Binary Search Trees
 - $O(\log N)$ access if balanced
 - $O(\log N)$ insertion if balanced
 - $O(\log N)$ deletion if balanced

Why are Binary Trees Better?

- ▶ Divide and Conquer - splitting problem into smaller problems
- ▶ Can we reduce the work by a bigger factor? 3? 10? More?
- ▶ An ArrayList does this in a way when *accessing* elements
 - *but must use an integer value*
 - *each position holds a single element*
 - ***given the index in an array, I can access that element rather quickly***
 - ***determining the address of the element requires a multiply op and an add op***

Hash Tables

- ▶ Hash Tables maintaining the fast access of arrays but improve the order for insertion, and deletion compare to array based lists.

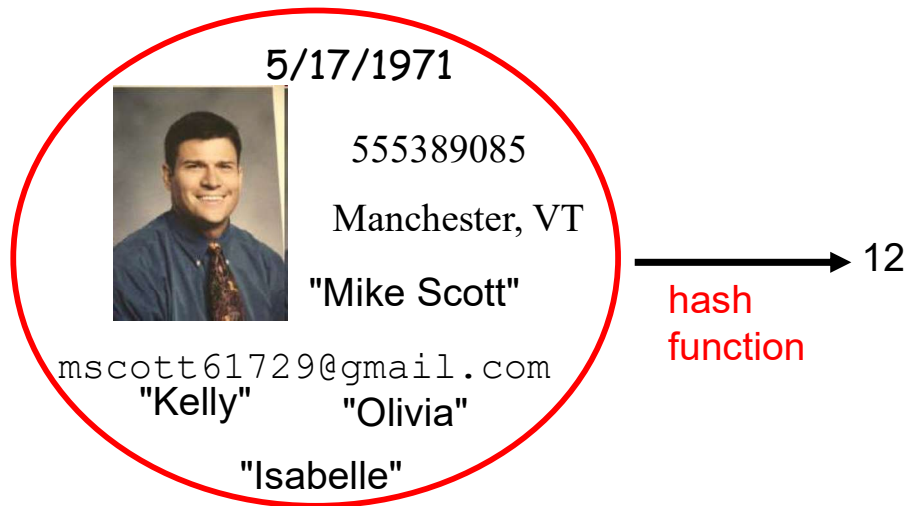


- ▶ Hash tables use an *array* and *hash functions* to determine the index for each element.

Hash Functions

- ▶ Hash: "From the French hacher, which means 'to chop'."
- ▶ *to hash* to mix randomly or shuffle (To cut up, to slash or hack about; to mangle)
- ▶ Hash Function: Take a piece of data and transforms it to a different piece of data (typically smaller), usually a single integer.
 - A function or algorithm
 - The input need not be integers!

Hash Function



Hash Functions

- ▶ Like a fingerprint

The screenshot shows the Apache OpenOffice download page. It includes a green header with the text "Download Apache OpenOffice" and "(Hosted by SourceForge.net - A trusted website)". Below this, it says "Select your favorite operating system, language and version:" and shows three dropdown menus: "Windows (EXE)", "English [US]", and "4.1.7". There are two buttons: "Download full installation" and "Download language pack". At the bottom, there is a red box highlighting the following text: "Full installation: File size: 134 MByte | Signatures and hashes: KEYS, ASC, SHA256, SHA512" and "Language pack: File size: 18 MByte | Signatures and hashes: KEYS, ASC, SHA256, SHA512".

- ▶ 134 Megabytes

Hash Function

- ▶ SHA 512 Hash code

```
c3e6ef13fb80e349813047547174a35367fb19ee
0ba704a27ab748ed99a6463671bcd1bef348bb42
a5e883301ef786970fe303d0fd3f677f8d0a8fd0
4aeba798
*Apache_OpenOffice_4.1.7_Win_x86_install
_en-US.exe
```

Simple Example

- ▶ Assume we are using names as our key
 - take 3rd letter of name, take int value of letter ($a = 0, b = 1, \dots$), divide by 6 and take remainder
- ▶ What does "Bellers" hash to?
- ▶ L -> 11 -> $11 \% 6 = 5$

Result of Hash Function

- ▶ Mike = $(10 \% 6) = 4$
- ▶ Kelly = $(11 \% 6) = 5$
- ▶ Olivia = $(8 \% 6) = 2$
- ▶ Isabelle = $(0 \% 6) = 0$
- ▶ David = $(21 \% 6) = 3$
- ▶ Margaret = $(17 \% 6) = 5$ (uh oh)
- ▶ Wendy = $(13 \% 6) = 1$
- ▶ This is an imperfect hash function. A perfect hash function yields a one to one mapping from the keys to the hash values.
- ▶ What is the maximum number of values this function can hash perfectly?

Clicker 3 - Hash Function

- ▶ Assume the hash function for String adds up the Unicode value for each character.

```
public int hashCode(String s) {  
    int result = 0;  
    for (int i = 0; i < s.length(); i++)  
        result += s.charAt(i);  
    return result;  
}
```

- ▶ Hashcode for "DAB" and "BAD"?

- A. 301 103
- B. 4 4
- C. 412 214
- D. 5 5
- E. 199 199

More on Hash Functions

- ▶ transform the key (which may not be an integer) into an integer value
- ▶ The transformation can use one of four techniques
 - Mapping
 - Folding
 - Shifting
 - Casting

Hashing Techniques

- ▶ Mapping
 - As seen in the example
 - integer values or things that can be easily converted to integer values in key
- ▶ Folding
 - partition key into several parts and the integer values for the various parts are combined
 - the parts may be hashed first
 - combine using addition, multiplication, shifting, logical exclusive OR

Shifting

- More complicated with shifting

```
int hashVal = 0;
int i = str.length() - 1;
while(i > 0)
{ hashVal = (hashVal << 1) + (int) str.charAt(i);
  i--;
}
```

different answers for "dog" and "god"

Shifting may give a better range of hash values when compared to just folding

Casts

- Very simple
 - essentially casting as part of fold and shift when working with chars.

The Java String class hashCode method

```
public int hashCode() {
    int h = hash;
    if (h == 0 && value.length > 0) {
        char[] val = value;
        for (int i = 0; i < val.length; i++) {
            h = 31 * h + val[i];
        }
        hash = h;
    }
    return h;
}
```



Mapping Results

- Transform hashed key value into a legal index in the hash table
- Hash table normally uses an array as its underlying storage container
- Normally get location on table by taking result of hash function, dividing by size of table, and taking remainder

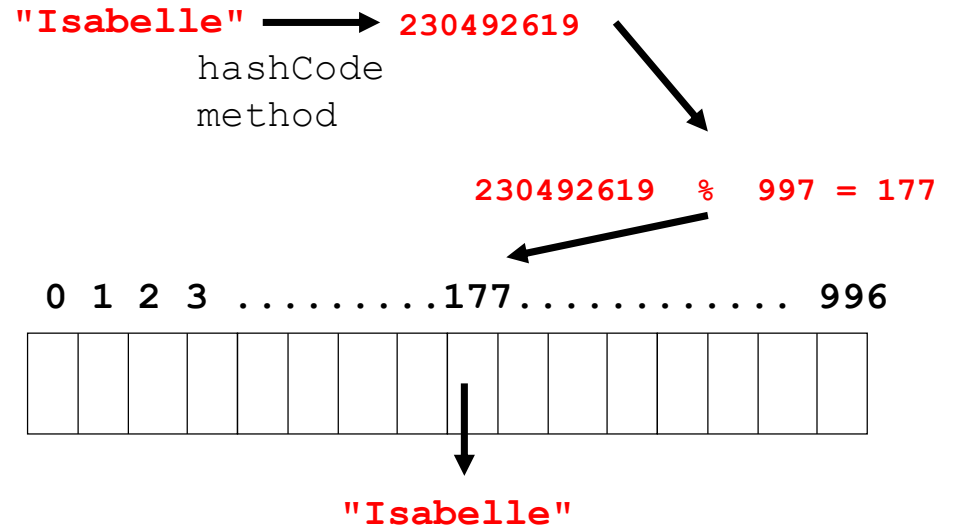
$$\text{index} = \text{key} \bmod n$$

n is size of hash table

empirical evidence shows a prime number is best

10 element hash table, move up to 11 or 13 elements

Mapping Results



Handling Collisions

- ▶ What to do when inserting an element and already something present?



CS314

Hash Tables

25

Open Addressing

- ▶ Could search forward or backwards for an open space
- ▶ Linear probing:
 - move forward 1 spot. Open?, 2 spots, 3 spots
 - reach the end?
 - When removing, insert a blank
 - null if never occupied, blank if once occupied
- ▶ Quadratic probing
 - 1 spot, 2 spots, 4 spots, 8 spots, 16 spots
- ▶ Resize when *load factor* reaches some limit



CS314

Hash Tables

Closed Addressing: Chaining

- ▶ Each element of hash table be another data structure
 - linked list, balanced binary tree
 - More space, but somewhat easier
 - everything goes in its spot
- ▶ What happens when resizing?
 - Why don't things just collide again?



CS314

Hash Tables

Hash Tables in Java

- ▶ `hashCode` method in `Object`
- ▶ `hashCode` and `equals`
 - "If two objects are equal according to the `equals` (`Object`) method, then calling the `hashCode` method on each of the two objects must produce the same integer result."
 - if you override `equals` you need to override `hashCode`
- ▶ Overriding one of `equals` and `hashCode`, but not the other, can cause logic errors that are difficult to track down if objects added to hash tables.

CS314

Hash Tables

28

Hash Tables in Java

- ▶ HashSet class
 - implements Set interface with internal storage container that is a HashTable
 - compare to TreeSet class, internal storage container is a Red Black Tree
- ▶ HashMap class
 - implements the Map interface, internal storage container for keys is a hash table

Comparison

- ▶ Compare these data structures for speed:
- ▶ Java HashSet
- ▶ Java TreeSet
- ▶ our naïve Binary Search Tree
- ▶ our HashTable
- ▶ Insert random ints

Clicker 4

- ▶ What will be order from fastest to slowest?
- A. HashSet TreeSet HashTable314 BST
 - B. HashSet HashTable314 TreeSet BST
 - C. TreeSet HashSet BST HashTable314
 - D. HashTable314 HashSet BST TreeSet
 - E. None of these