Hashtable details
Sorting Revisited

Map/Hashtables

- Each item is a key, value pair
  - for example
    - Key — Value
    - studentId — studentRecord
    - town+state — Place
    - word — Definition
  - Sometimes a Map is called a Dictionary
- There is a Set of Keys
- And a Collection of Values
- Goal: efficient add and removal (no concern for order)

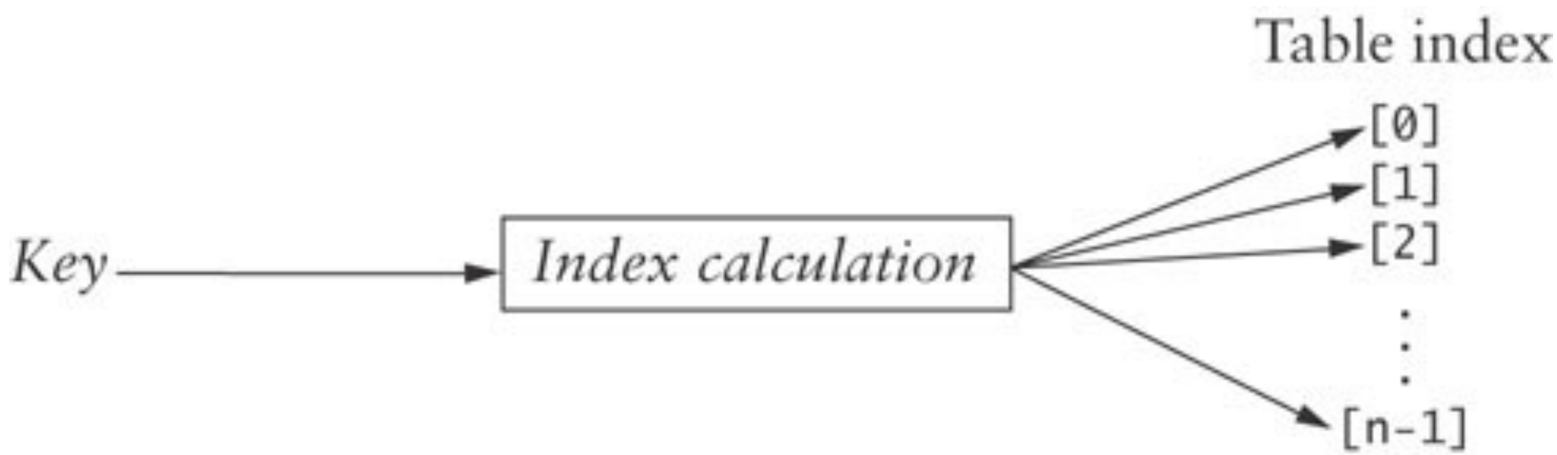
Map interface

- public V put(K key, V value)
- public V get(K key)
- public V remove(K key)

- The expected is O(1) for a Hashtable, but the worse case is O(n)
- A SortedHashMap has O(log n) insertion and removal

Hash Codes and Index Calculation

- The basis of hashing is to transform the item's key value into an integer value (its *hash code*) which is then transformed into a table index



+ Valid Keys

- Keys are typically
  - numbers
  - characters (also numbers)
  - Strings (sequences of characters, which are numbers)
- In Java, any Object can be a Key.
  - hashCode() can be overridden, and should be overridden if equals has been overridden.

Goal

- similar (really all) keys map to different locations
- Question:
  - How do we map a large number of possible values to a much smaller table size.
  - e.g. if we use just 10 letter strings of lowercase letters, then there are 26^{10} possible keys.
- Collisions cause the O(1) property to fail.

Potential Hashing functions

- F1: (letter1 + letter2 + letter 3 … + letter n) % tablesize
  - ignores order of letters
- F2 : use position and code for letter
- F3 : Java uses 31^(position) * code(letter)

Desirable properties

- codes generated are random
  - in int range
  - in int % tablesize range
- easy to compute

Handling Collisions

- open addressing
  - If there is a collision, check the next available spot until an open spot is found
  - linear probing
  - quadratic probing
- chaining
  - if there is a collision, add it to the bucket (a linked list)

+ Open Addressing Example (Processing Sketches)

Rhoads	1847927497	4	[0]
Thomas	1790657756	8	[1]
Park	2480138	8	[2]
McBride	1777921980	6	[3]
Wofford	1116374487	6	[4]
McPherson	64559159	8	[5]
Vickers	2116683019	1	[6]
McAuliffe	15268542	6	[7]
Cassidy	2075000160	3	[8]

Problem with Open Addressing

- removal
  - need to keep track of places that once held data

- The chaining strategy removes this concern.

Chaining

- When collide add to bucket (list)
- removal just remove the item at the hash location from the list

Chaining Example (on board)

Rhoads	1847927497	4	[0]	
Thomas	1790657756	8	[1]	
Park	2480138	8	[2]	
McBride	1777921980	6	[3]	
Wofford	1116374487	6	[4]	
McPherson	64559159	8	[5]	
Vickers	2116683019	1	[6]	
McAuliffe	15268542	6	[7]	
Cassidy	2075000160	3	[8]	

Performance of Hash tables

- load factor = #filled cells / table size
- lower load factor leads to better performance
- higher load factor leads to worse performance.

+ Performance of Hash Tables versus Sorted Array and Binary Search Tree

- The number of comparisons required for a binary search of a sorted array is O(log *n*)
  - A sorted array of size 128 requires up to 7 probes (2^7 is 128) which is more than for a hash table of any size that is 90% full
  - A binary search tree performs similarly
- Insertion or removal

hash table	O(1) expected; worst case O(*n*)
unsorted array	O(*n*)
binary search tree	O(log *n*); worst case O(*n*)

Questions?

Balancing a Binary search tree

- The key to all balanced tree methods is the concept of rotation
  - rotate left
  - rotate right

Sorting Review

- Selection Sort
- Insertion Sort
- Merge Sort