Elementary Symbol Tables

Reference: Chapter 12, Algorithms in Java, 3rd Edition, Robert Sedgewick.

Symbol Table ADT

Symbol table: key-value pair abstraction.

- Insert a value with specified key.
- Search for value given key.
- Delete value with given key.

DNS lookup.

- Insert URL with specified IP address.
- Given URL, find corresponding IP address.

URL	IP address
www.cs.princeton.edu	128.112.136.11
www.princeton.edu	128.112.128.15
www.yale.edu	130.132.143.21
www.harvard.edu	128.103.060.55
www.simpsons.com	209.052.165.60
key	value

Symbol Table Applications

Application	Purpose	Key	Value
Phone book	Look up phone number	Name	Phone number
Bank	Process transaction	Account number	Transaction details
File share	Find song to download	Name of song	Computer ID
File system	Find file on disk	Filename	Location on disk
Dictionary	Look up word	Word	Definition
Web search	Find relevant documents	Keyword	List of documents
Book index	Find relevant pages	Keyword	List of pages
Web cache	Download	Filename	File contents
Genomics	Find markers	DNA string	Known positions
DNS	Find IP address given URL	URL	IP address
Reverse DNS	Find URL given IP address	IP address	URL
Compiler	Find properties of variable	Variable name	Value and type
Routing table	Route Internet packets	Destination	Best route

Symbol Table Client: DNS Lookup

DNS lookup client program.

- `st.put(key, value)` inserts a key-value pair into symbol table.
- `st.get(key)` searches for the given key and returns the value.

```
public static void main(String[] args) {
   ST<String, String> st = new ST<String, String>();

   st.put("www.cs.princeton.edu", "128.112.136.11");
   st.put("www.princeton.edu",    "128.112.128.15");
   st.put("www.yale.edu",         "130.132.143.21");
   st.put("www.simpsons.com",     "209.052.165.60");
      ↑
  st["www.simpsons.com"] = "209.052.165.60"

   System.out.println(st.get("www.cs.princeton.edu"));
   System.out.println(st.get("www.harvardsucks.com"));
   System.out.println(st.get("www.simpsons.com"));
}                             ↑
                        st["www.simpsons.com"]
```

```
128.112.136.11
null
209.052.165.60
```

Symbol Table Client: Frequency Counter

Frequency counter. [e.g., web traffic analysis]

- Read in a key.
- If key is in symbol table, increment counter by one;
  If key is not in symbol table, insert it with count = 1.

```
public static void main(String args[]) {
   ST<String, Integer> st = new ST<String, Integer>();

   while (!StdIn.isEmpty()) {
      String key = StdIn.readString();
      if (st.contains(key))
         st.put(key, st.get(key) + 1);
      else
         st.put(key, 1);
   }
```

calculate frequencies

print results

```
   for (String s : st)
      System.out.println(st.get(s) + " " + s);
}
```

Symbol Table Interface

Symbol table interface.

put(key, value)	insert the key-value pair
get(key)	return value associated with given key
remove(key)	remove the key
contains(key)	is given key present?
iterator()	return iterator over all keys

Convention.

- No duplicate keys. [Old values overwritten with new ones.]
- Values are not null.

Associative array. Unique value associated with each key.

Q. How to implement a lazy remove?

Keys and Values

Key type.

- Some implementations assumes keys have `compareTo` method.
- Other implementations assume keys have `equals` and `hashCode` methods.

Value type. Any generic type.

```
private static boolean less(Comparable v, Comparable w) {
   return v.compareTo(w) < 0;
}

private static boolean eq(Comparable v, Comparable w) {
   return v.compareTo(w) == 0;
}
```

helper functions
(when Key is Comparable)

Symbol Table: Sorted Array Implementation

Maintain array of keys and values.

- Store in sorted order by key.
- `keys[i]` = ith largest key.
- `vals[i]` = value associated with ith largest key.

```
public class SortedST<Key extends Comparable, Value> {
implements Iterable<Key>
   private Object[]     vals;
   private Comparable[] keys;
   private int N;
```

4	6	14	20	26	32	47	55	56	58	82	▨	▨	▨

Symbol Table Search: Sorted Array Implementation

Binary search.

- Examine the middle key.
- If it matches, return the value.
- Otherwise, search either the left or right half.

```
public Value get(Key key) {
   int l = 0;
   int r = N-1;
   while (l <= r) {
      int m = (l + r) / 2;
      if (  eq(key, keys[m])) return vals[m];
      if (less(key, keys[m])) r = m - 1;
      else                    l = m + 1;
   }
   return null;
}
```

Symbol Table Insert: Sorted Array Implementation

Insert.

- Need to maintain entries in ascending order.
- Find insertion point and move larger keys to the right.

4	6	14	20	26	32	47	55	56	58	82	▨	▨	key = value = int
4	6	14	20	26	28	32	47	55	56	58	82	▨	Insert 28

Performance Cost Summary

	Worst Case		Average Case	
Implementation	Search	Insert	Search	Insert
Sorted array	log N	N	log N	N / 2

Sorted array. Fast search, slow insert.

Equals

Equivalence relation. For any references `x`, `y` and `z`:

- Reflexive: `x.equals(x)` is `true`.
- Symmetric: `x.equals(y)` iff `y.equals(x)`.
- Transitive: if `x.equals(y)` and `y.equals(z)`, then `x.equals(z)`.
- Consistency: multiple invocations of `x.equals(y)` return the same value, provided neither changes between invocations.
- Non-null: `x.equals(null)` is `false`.

Default implementation: `(x == y)`.
Customized implementations: `String, URL, Integer`.

Best practices. If class is `Comparable`, make `equals` consistent with `compareTo`.

Immutable Keys

Best practices. Use immutable types as keys.

- Immutable in Java: `String`, `Integer`, `BigInteger`.
- Mutable in Java: `Date`, `GregorianCalendar`.

> "Note: great care must be exercised if mutable objects are used as map keys. The behavior of a map is not specified if the value of an object is changed in a manner that affects equals comparisons while the object is a key in the map. A special case of this prohibition is that it is not permissible for a map to contain itself as a key. " - *Sun JavaDoc for Map interface*

Implementing Equals: US Phone Numbers

Phone numbers: (609) 867-5309.

exchange

area code

extension

final helps enforce immutability

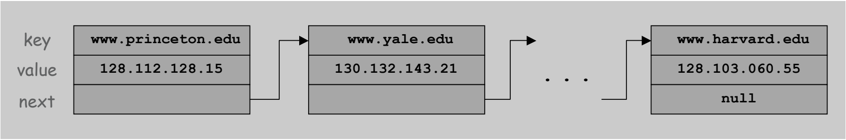
```
public final class PhoneNumber {
   private final int area, exch, ext;

   public PhoneNumber(int area, int exch, int ext) {
      this.area = area;
      this.exch = exch;
      this.ext  = ext;
   }

   public boolean equals(Object y) {
      if (y == this) return true;
      if (y == null) return false;
      if (y.getClass() != this.getClass()) return false;
      PhoneNumber a = this;
      PhoneNumber b = (PhoneNumber) y;
      return (a.area == b.area) && (a.exch == b.exch)
                                && (a.ext == b.ext);
   }
}
```

can't throw an exception

Symbol Table: Linked List Implementation

Maintain a linked list of key-value pairs.

- Insert new key-value pair at beginning of list.
- Use exhaustive search to search for a key.



Symbol Table: Linked List Implementation

```
public class ListST<Key, Value> implements Iterable<Key> {
   private Node first;

   private class Node {
      Key   key;
      Value val;
      Node  next;
      Node(Key key, Value val, Node next)  {
         this.key  = key;
         this.val  = val;
         this.next = next;
      }
   }

   public Iterator<Key> iterator() {
      return new ListIterator();
   }
```

similar to Sequence iterator

Symbol Table: Linked List Implementation (cont)

```
   public Value get(Key key) {
      for (Node x = first; x != null; x = x.next)
         if (key.equals(x.key))
            return x.val;
      return null;
   }

   public void put(Key key, Value val) {
      for (Node x = first; x != null; x = x.next) {
         if (key.equals(x.key)) {
            x.val = val;
            return;
         }
      }
      first = new Node(key, val, first);
   }
}
```

Performance Cost Summary

	Worst Case		Average Case	
Implementation	Search	Insert	Search	Insert
Sorted array	log N	N	log N	N / 2
Unsorted list	N	N	N / 2	N

Sorted array. Fast search, slow insert.
Linked list. Slow insert, slow search.

Q. Can we achieve O(log N) for all ops?