Symbol Tables

Symbol tables

Ordered array

Unordered linked list

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. ⬅ In this lecture, we assume key is a String.
- Search for value given key.
- Delete value with given key.

Example: key = URL, value = IP address.

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

Web Site	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

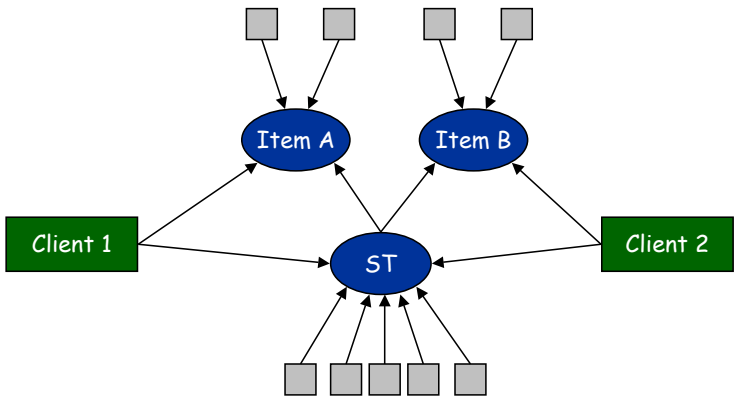
Other Symbol Table Applications

Other applications.

- Online phone book: look up a name to find telephone number.
- Google: look up phrase and return most relevant web pages.
- Spell checker: look up a word to find if it's there or present alternative.
- ➡ DNS: look up name of web site to find IP address.
- Java compiler: look up variable name to find its type and value.
- File sharer: look up song to find host machines.
- File system: look up file name to find location on hard drive.
- University registrar: look up student to find grades.
- Web traffic analyzer: look up host to find number of hits.
- Web cache: cache frequently accessed pages.
- Routing table: look up routing info for IP.
- Browser: highlight visited links in purple.
- Chess: detect a repetition draw.
- Bayesian spam filter: use frequencies of spam and ham words to filter email.
- Language modeling: determine frequency of next letter given prefix.
- Book index: determine pages on which each word appears.
- "Associative memory."
- Index of any kind.
- . . .

Abstract Data Types

Interface. Description of data type, basic ops.
Client. Program using ops defined in interface.
Implementation. Actual code implementing ops.



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) {
   SymbolTable st = new SymbolTable();
                    key                          value
   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: Remove Duplicates

Remove duplicates (e.g., from commercial mailing list).

- Read in a key.
- If key is not in symbol table, print out key and insert.

```
public class DeDup {
   public static void main(String[] args) {
      SymbolTable st = new SymbolTable();
      while (!StdIn.isEmpty()) {
         String key = StdIn.readString();
         if (st.get(key) == null) {
            System.out.println(key);
            st.put(key, "");
         }              insert empty string as value
      }
   }
}
```


Object

Class `Object`.

- All objects "inherit" from the special class `Object`.
- All objects have certain pre-defined methods.

Method	Description	Default	Typical Usage
`toString`	convert to string	memory address	`"hello " + s`
`equals`	are two objects equal?	are two memory addresses equal?	`if (s.equals(t))`
`hashCode`	convert to integer	memory address	`s.hashCode()`

Consequences.

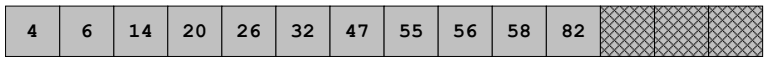
- Can have a symbol table of any object, e.g, `String` or `Student`.
- Programmer may need to override default methods.

Symbol Table: Sorted Array Implementation

Maintain array of keys and values.

- Store in sorted order by key.
- `keys[i]` = i^{th} largest key.
- `values[i]` = value associated with i^{th} largest key.

```
public class ST {
   private Object[] values = new Object[14];
   private String[] keys   = new Object[14];
   private int N = 0;  number of elements    initial capacity
```



Symbol Table Search: Sorted Array Implementation

Binary search.

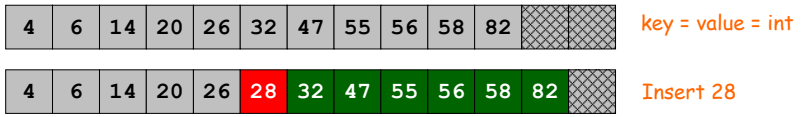
- Examine the middle key.
- If it matches, return the value.
- Otherwise, search either the left or right half.

```
public Object get(String key) {
   int left  = 0;
   int right = N-1;
   while (left <= right) {
      int mid = (left + right) / 2;
      if (equal(key, keys[mid])) return values[mid];   found
      if (less (key, keys[mid])) right = mid - 1;      left half
      else                       left  = mid + 1;      right half
   }
   return null;   not found
}
```


Symbol Table Insert: Sorted Array Implementation

Insert.

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



Sorted Array Implementation: Performance

Advantages: not much code, fast search.

```
% java DeDup < toSpamList.txt
wayne@cs.princeton.edu
rs@cs.princeton.edu
dgabai@cs.princeton.edu
pcalamia@cs.princeton.edu
sgaw@cs.princeton.edu
```

```
% java Dedup < mobydick.txt
moby
dick
herman
melville
call
me
ishmael
some
years
ago                  210,028 words
. . .                16,834 distinct
```

Disadvantage: insert is hopelessly slow for large inputs.

⬆

hours to dedup Moby Dick

Sorted Array Implementation Analysis

Claim. Worst-case number comparisons to binary search in a sorted array of size N is $O(\log N)$?

- Divide list in half each time. $625 \Rightarrow 312 \Rightarrow 156 \Rightarrow 78 \Rightarrow 39 \Rightarrow 18 \Rightarrow 9 \Rightarrow 4 \Rightarrow 2 \Rightarrow 1$

Proof. Worst-case number of steps satisfies:

- $C(N) = 1 + C(N/2)$ (integer division)
- $C(1) = 0$

  $\Rightarrow C(N) = \lceil \log_2 (N+1) \rceil$

$625_{10} = 1001110001_2$

$\lceil \log_2 (626) \rceil = 10$

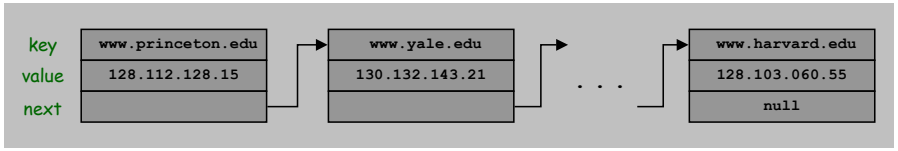
- Same recurrence as # bits in binary representation of N.

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

Symbol Table: Linked List Implementation

Maintain a linked list of key-value pairs.

- Insert new key-value pair at beginning of list.
- Key = `String`, value = `Object`.
- Use exhaustive search to search for a key.



Symbol Table: Linked List Implementation

```
public class SymbolTable {
   private List st;          ⇐ a linked list of key-value pairs

   private static class List {
      String key;
      Object value;
      List next;
      List(String k, Object val, List next) {
         this.key   = k;
         this.value = val;
         this.next  = next;
      }
   }                                  helper inner class for linked lists

   public void put(String k, Object val) {
      st = new List(k, val, st);
   }                                  insert at front of list

   public Object get(String k) {
      for (List x = st; x != null; x = x.next)
         if (k.equals(x.key)) return x.value;
      return null;
   }              ↖ not found        exhaustively search for key
}
```


Linked List Implementation: Performance

Advantages: not much code, fast insertion.

```
% java DeDup < toSpamList.txt
wayne@cs.princeton.edu
rs@cs.princeton.edu
dgabai@cs.princeton.edu
pcalamia@cs.princeton.edu
sgaw@cs.princeton.edu
```

```
% java Dedup < mobydick.txt
moby
dick
herman
melville
call
me
ishmael
some
years
ago                    210,028 words
. . .                  16,834 distinct
```

Disadvantage: search is hopelessly slow for large inputs.

⬆ hours to dedup Moby Dick

Linked List Implementation: Analysis

Insertion. Constant time.

Search. Need to look at every entry if not found.

	Worst Case			Average Case		
Implementation	Search	Insert	Delete	Search	Insert	Delete
Sorted array	log N	N	N	log N	N / 2	N / 2
Unsorted list	N	1	1	N / 2	1	1

⬆ given reference to element to be deleted

Can we achieve log N performance for all ops?