



## Symbol Table Client: Remove Duplicates

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

- Read in a key.
- If key is already in the symbol table, do nothing.
- Otherwise, print out key and insert it into symbol table.

```
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, "");
            }
        }
    }
}
```

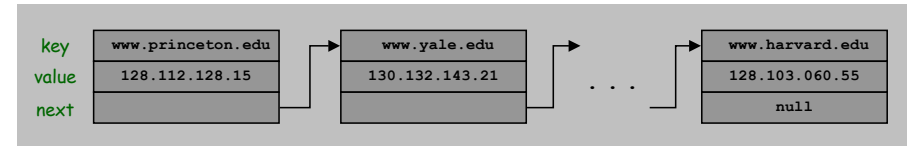
↑ insert empty string as value

5

## 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.



6

## Symbol Table: Linked List Implementation

```
public class SymbolTable {
    private List st; // ← a linked list of (key, value) pairs

    private class List {
        String key;
        Object val;
        List next;
        List(String key, Object val, List next) {
            this.key = key;
            this.val = val;
            this.next = next;
        }
    }

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

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

7

## 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 < moby dick.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.

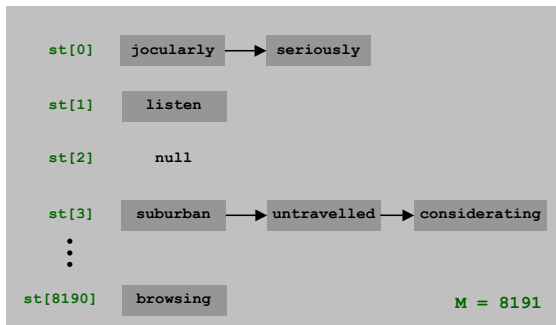
↑ hours to dedup Moby Dick

8

## Hashing

### Hashing.

- Goal: speed up search by a factor of  $M$  by making lists shorter.
- Array  $st$  of  $M$  linked lists (or chains).
- Map from string key to integer  $i$  between  $0$  and  $M-1$ .
  - put key-value pair in  $i^{\text{th}}$  linked list



key	hash
call	7121
me	3480
ishmael	5017
seriously	0
untravelled	3
suburban	3
...	.

9

## Choosing a Good Hash Function

### Goal: scramble the keys.

- Each table position **equally likely** for each key.
  - ↳ thoroughly researched problem

### Ex: Social Security numbers.

- Bad: first three digits. ← 573 = California, 574 = Alaska
- Better: last three digits. ← assigned in chronological order within a given geographic region

### Ex: Strings.

- Bad: first few letters (converted to an int).
- Good: do calculation involving all characters

```
public int hashCode() {
    int hash = 0;
    for (int i = 0; i < length(); i++)
        hash = (31 * hash) + charAt(i);
    return hash;
}
String.java
```

```
s = "call";
h = s.hashCode();
hash = h % M;
7121 8191 3045982
insert call into chain 7121
```

10

## Symbol Table: Hash Table Implementation

```
public class SymbolTable {
    private int M = 8191; // number of chains (should be prime)
    private List[] st = new List[M];

    private class List { AS BEFORE }

    public static int hash(String s) {
        return Math.abs(s.hashCode()) % M;
        // between 0 and M-1
    }

    void put(String key, Object val) {
        int i = hash(key, M);
        st[i] = new List(key, val, st[i]);
        // insert at front of i-th chain
    }

    Object get(String key) {
        int i = hash(key, M);
        for (List x = st[i]; x != null; x = x.next)
            if (key.equals(x.key)) return x.val;
        return null;
        // exhaustively search i-th chain for key
    }
}
```

11

## Hash Table Implementation: Performance

Advantages: fast insertion, fast search.

Disadvantage: hash table has fixed size.

↳ easily corrected

Hash tables improves ALL symbol table clients.

- Makes difference between practical solution and no solution.
- Ex: Moby Dick now takes a few seconds instead of hours.

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

12

## Question

Current code searches for an IP address given a URL.

- "DNS lookup."

What if we want to search for a URL given an IP address?

- "Reverse DNS lookup."



13

## Symbol Table Summary

Symbol table: quintessential database lookup data type.

Different performance characteristics with different implementations.

- Linked list, hash table, ...
- Java has built-in libraries for symbol tables.
  - HashMap = hash table implementation.
  - TreeMap = *red-black* tree implementation.

```
import java.util.HashMap;
public class HashMapDemo {
    public static void main(String[] args) {
        HashMap st = new HashMap();
        st.put("www.cs.princeton.edu", "128.112.136.11");
        st.put("www.princeton.edu", "128.112.128.15");
        st.put("www.simpsons.com", "209.052.165.60");
        System.out.println(st.get("www.cs.princeton.edu"));
    }
}
```

15

## Summary

ADTs enable modular programming.

- Split program into smaller modules.
- Separate compilation.
- Different clients can share the same ADT.

ADTs enable encapsulation.

- Keep modules independent (include main() in class for testing).
- Can substitute different classes that implement same interface.
- No need to change client.

Issues of ADT design.

- Feature creep.
- Formal specification problem.

16