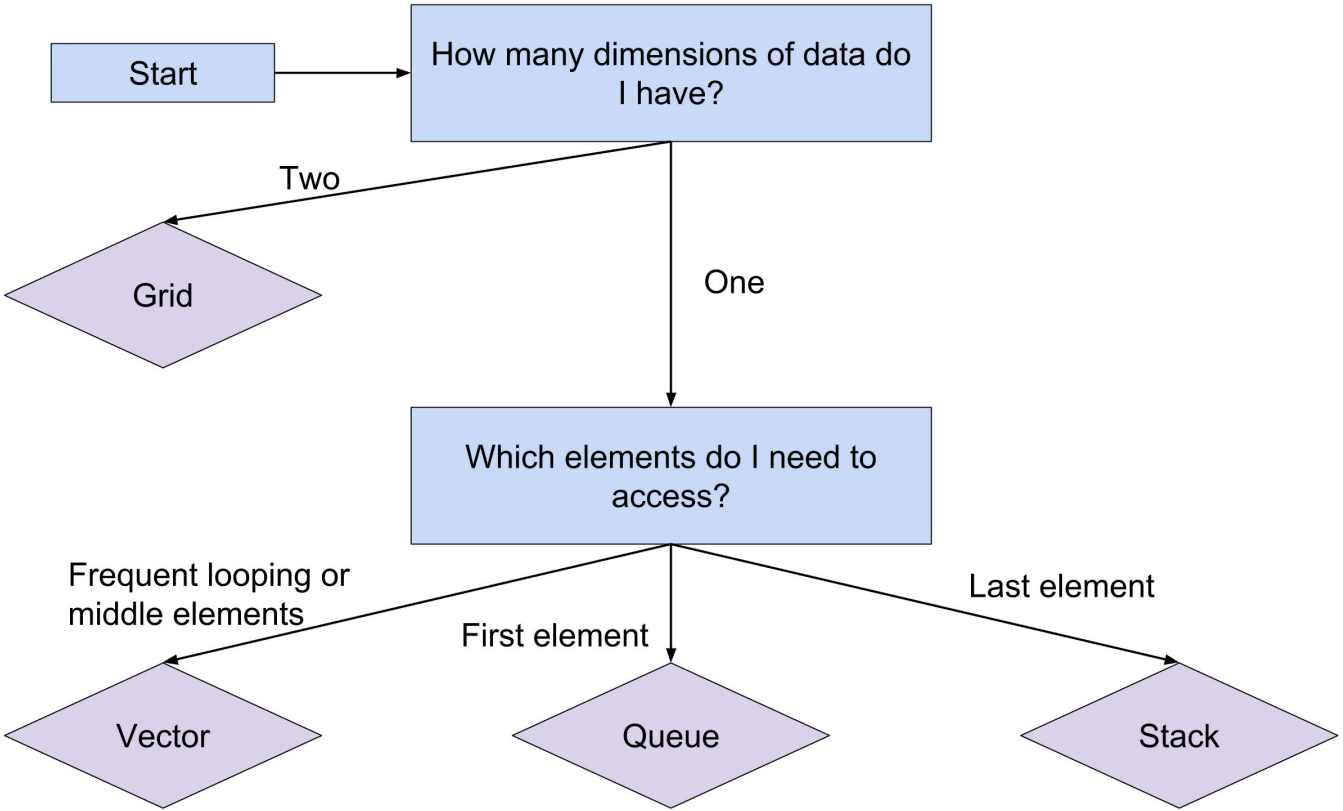
CS 106B, Lecture 7
Sets and Maps

ADTs So Far

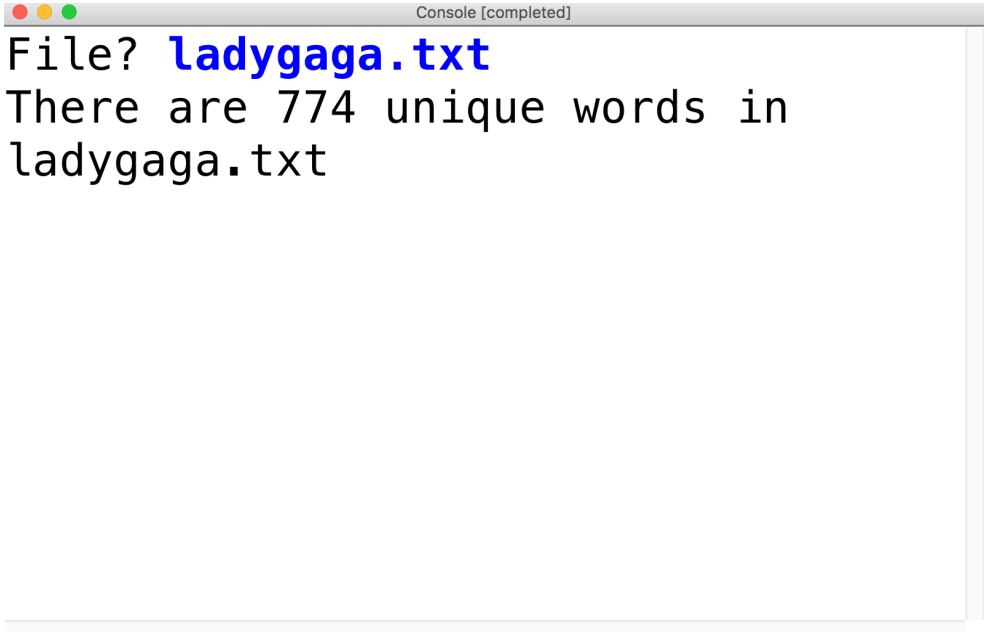

Start
How many dimensions of data do I have?
Two
Grid
One
Which elements do I need to access?
Frequent looping or middle elements
First element
Last element
Vector
Queue
Stack

Today's Topics

- Sets (no duplicates allowed!)
  - Lexicons
- Maps (map a key to a value)

CountUniqueWords

- One basic statistic about a text is the number of unique words it has
  - Linguists and computer scientists frequently start analysis with the number of unique words
  - Good indication of vocabulary
- Problem: how can we determine the number of unique words in a file?

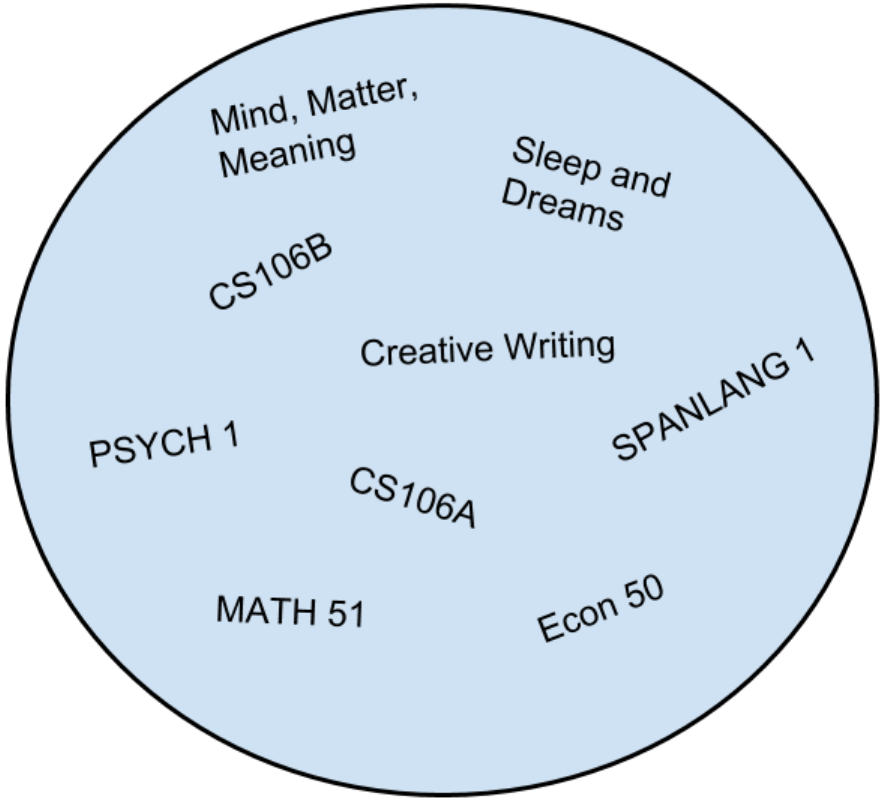


Today's Topics

- Sets (no duplicates allowed!)
  - Lexicons
- Maps (map a key to a value)

Sets

- Only answers question of membership
  - **No duplicates**
- Operations
  - `contains(`*`elem`*`)`
  - `add(`*`elem`*`)`
  - `remove(`*`elem`*`)`
- Comparison to `Vector`
  - Does not maintain insertion order
  - No duplicates
  - Really fast at finding membership

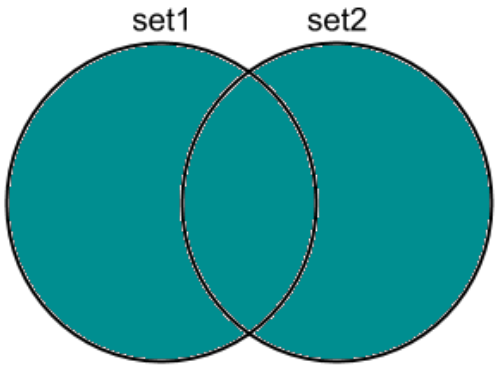


Looping over Sets

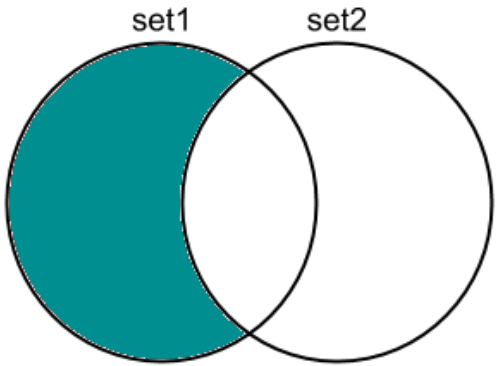
- Sets don't have indices, so we use a for-each loop
- Iterates in sorted order (alphabetical order for strings)
- Can't edit while we iterate

```
Set<string> friends;
friends.add("Leland");
friends.add("Kate");

// prints in alphabetical order
for (string myFriend : friends) {
      cout << "Hi, " << myFriend << endl;
      cout << "Let's get dinner." << endl;
}
```

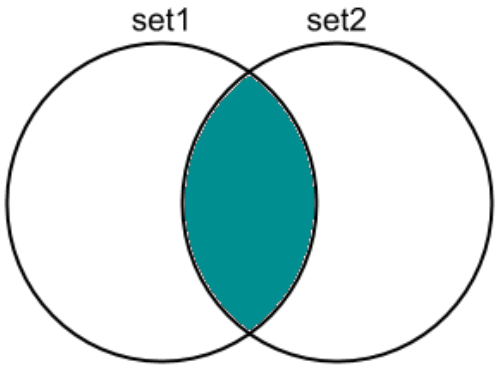
Good Operations to Know



Union: set1 + set2



Difference: set1 - set2



Intersection: set1 * set2

Sets – Method List

`s.add(value)`	O(log N)	Adds an element to this set, if it was not already there
`s.clear()`	O(N)	Removes all elements from this set
`s.contains(value)`	O(log N)	Returns **`true`** if ***value*** is in this set
`s.equals(set)`	O(N)	Returns **`true`** if the two sets contain the same elements
`s.first()`	O(log N)	Returns the first value in the set in order
`s.isEmpty()`	O(1)	Returns **`true`** if the set contains no elements
`s.isSubsetOf(s2)`	O(N)	Returns **`true`** if all the elements in the set are also in ***s2***
`s.remove(value)`	O(log N)	Removes an element from this set
`s.size()`	O(1)	Returns the number of elements in this set
`s.toString()`	O(N)	Converts the set to a printable string representation

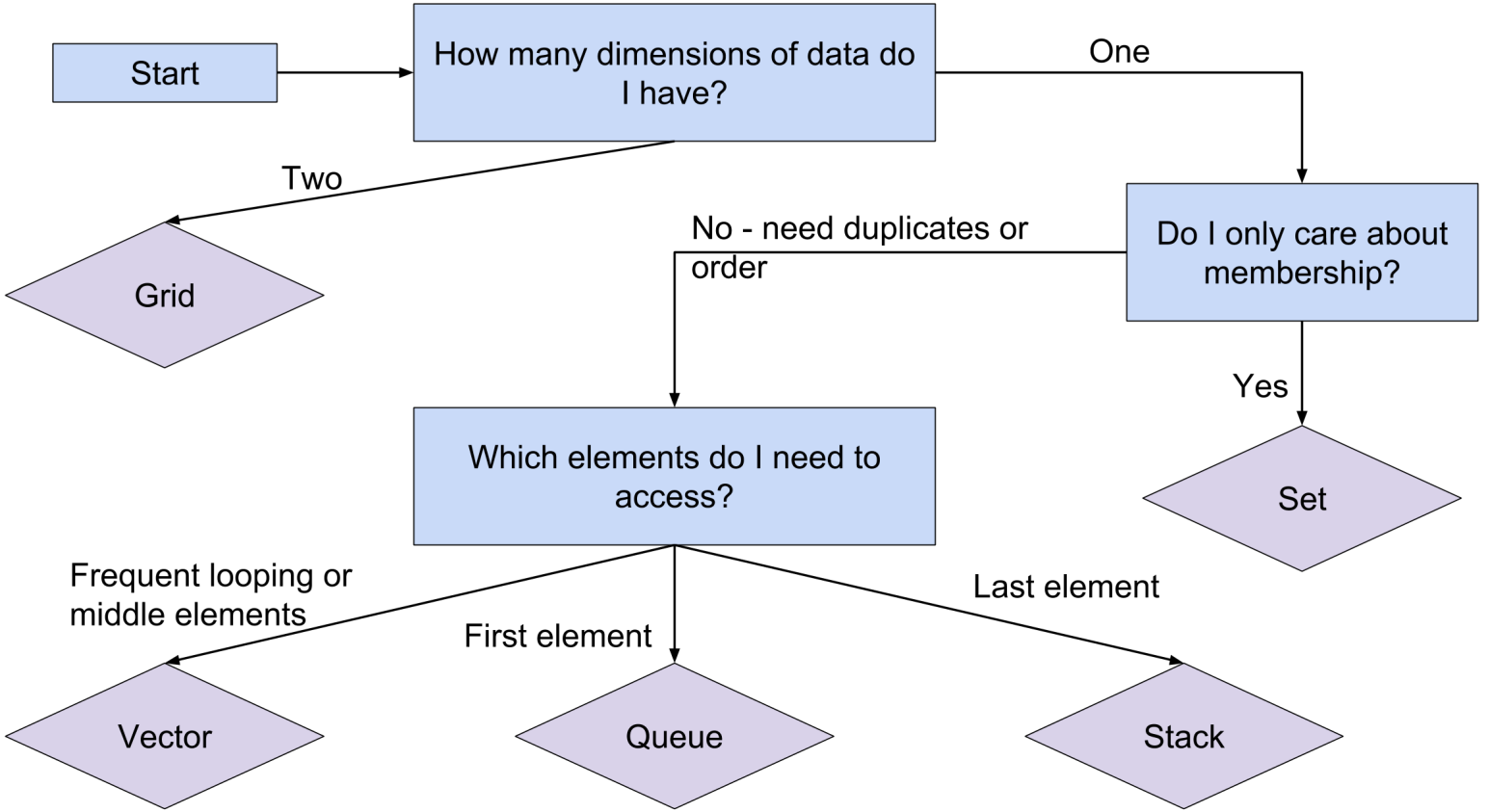
Today's Topics

- Sets (no duplicates allowed!)
  - Lexicons
- Maps (map a key to a value)

Lexicons

- Set where the only type is string
- Can do everything a Set does
- Also answers the question – do any words start with this prefix?
  - `lexicon.containsPrefix(prefix)`
- Used to store dictionaries
- We'll talk about lexicons more later

ADTs Expanded

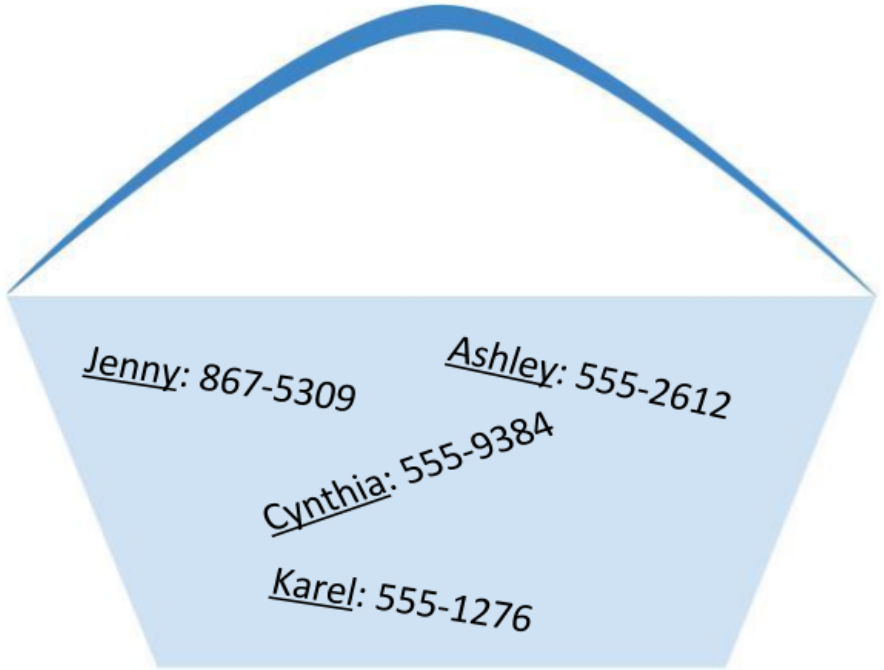


Today's Topics

- Sets (no duplicates allowed!)
  - Lexicons
- Maps (map a key to a value)

Maps

- Stores **pairs** of information
  - First half of the pair is called a **key**, and the second half is the associated **value**
  - Find a value by looking up its associated key
  - Keys must be unique (just like elements in a Set!)
- Comparison with Vector
  - Vectors look up elements by *index*, Maps look them up by *key*
  - Need to declare two types (for the key and the value)
  - Ordered by key, not index



Map Syntax

- *map*.put(*key*, *value*)
  - *map*[*key*] = *value*
  - Adds the key if it wasn't already in the map
  - Otherwise edits its value
- *map*.get(*key*)
  - *map*[*key*]
    - This alternate syntax will create a key with the **default** value in the map
- *map*.remove(*key*)
  - No effect if the key isn't in the map

Map Example: Dictionary

```
ifstream file;
promptUserForFile(file, "Where is your dictionary?");
Map<string, string> dictionary;
string word;

while (getline(file, word)) {
        string definition;
        getline(file, definition);
        dictionary[word] = definition;
}

while (true) {
        string query = getLine("Word to look up?");
        if (dictionary.containsKey(query)) {
                cout << "The definition is " << dictionary[query] << endl;
        } else {
                cout << "I don't know that word!" << endl;
        }
}
```

Looping over Maps

- Maps also don't have indices, so we use a for-each loop over the keys
- Iterates in sorted order over the keys
- Can't edit the keys while we iterate (can edit values)

```
Map<string, int> phonebook;
phonebook["Tyler"] = 5551234;
phonebook["Kate"] = 5559876;

// prints in alphabetical order
for (string name: phonebook) {
        int phoneNumber = phonebook[name];
        cout << "I'm going to call " << name;
        cout << " at " << phoneNumber << endl;
}
```

Word Count

- We've found the number of unique words in a file. Another statistic is how frequently each word is used.
- Given a text file and a user-inputted word, how frequently is that word used in the file?

```
to be or not to be
```

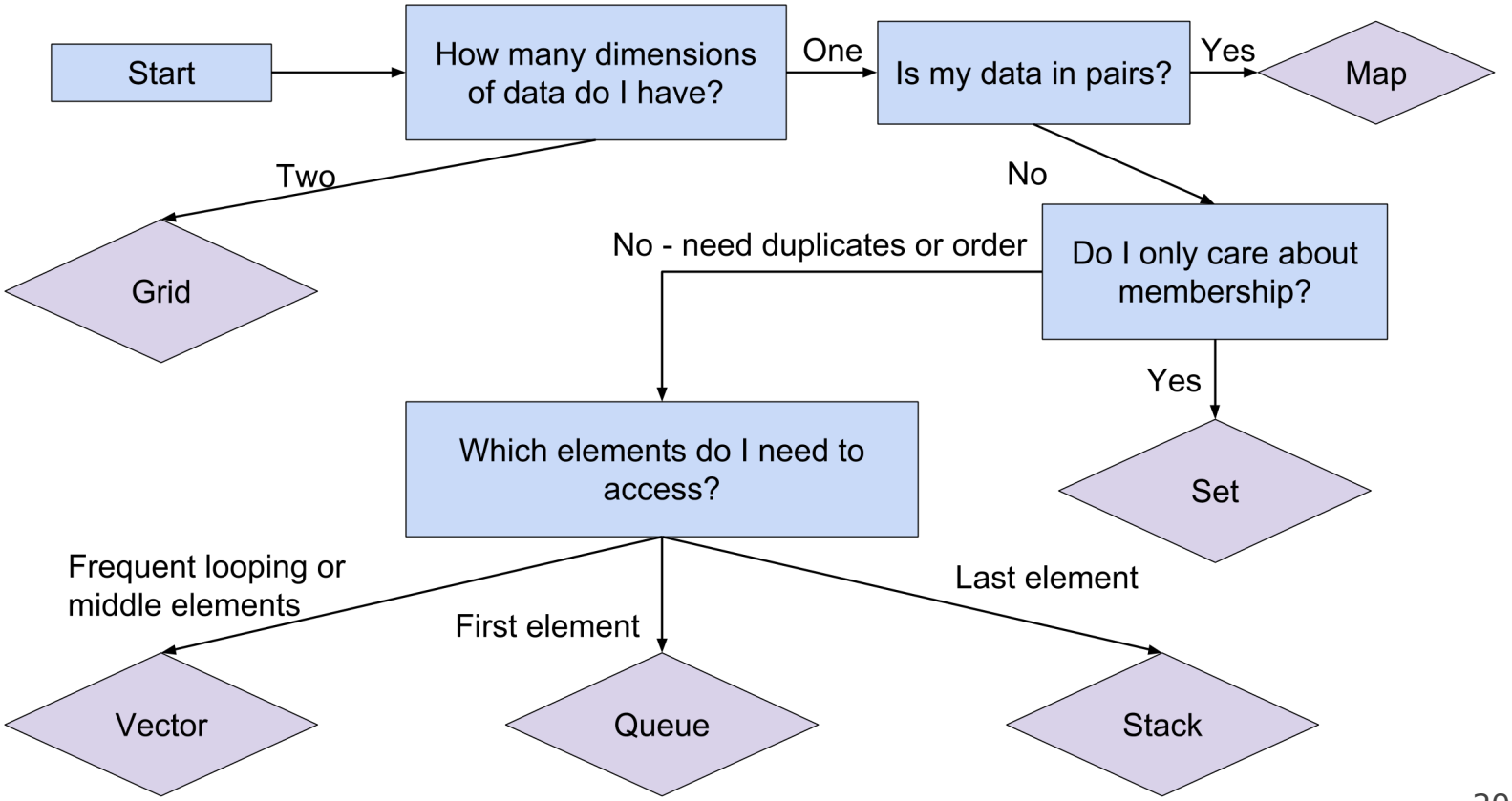
tiny.txt

```
File? tiny.txt

Word? to
"to" appears 2 times

Word? or
"or" appears 1 times
```

Solution

```
int main() {
    ifstream infile;
    promptUserForFile(infile, "File?");
    Map<string, int> map;
    string word;
    while (infile >> word) {
        map[word]++;
    }
    infile.close();

    string userWord = getLine("Enter a word (or enter to quit): ");
    while (userWord != "") {
        if (map.containsKey(userWord)) {
            cout << "\"" << userWord << "\"" << " appears " << map[userWord] << " times."
<< endl;
        }
        userWord = getLine("Enter a word (or enter to quit): ");
    }
    return 0;
}
```

ADT Soup


Start
How many dimensions of data do I have?
One
Is my data in pairs?
Yes
Map
Two
Grid
No
No - need duplicates or order
Do I only care about membership?
Yes
Set
Which elements do I need to access?
Frequent looping or middle elements
Vector
First element
Queue
Last element
Stack

Nesting ADTs: Where2Eat

- Problem: we want to schedule a dinner with some group of our friends
- We have a text file with all our friends' dinner preferences
- Given a group of friends going to a dinner, where should we eat to maximize happiness?
  - We might not be able to find a place that makes everyone happy – such is life

```
Ashley
In n Out
Chipotle
Axe and Palm

Ketan
Chipotle
Bytes Cafe

Karel
Bytes Café
Forbes Cafe
```

- Which ADT(s) should we use?

Solution

```
int main() {
    ifstream file;
    file.open("restaurants.txt");
    Map<string, Set<string> > map;

    string myFriend;
    while (getline(file, myFriend)) {
        string restaurant;
        while (getline(file, restaurant) &&
restaurant != "") {
            map[myFriend] += restaurant;
        }
    }
    Set<string> guests;
    string guest = getLine("Enter a guest: ");
```

```
while (guest != "") {

        while (guest != "") {
            guests.add(guest);
            guest = getLine("Enter a guest: ");
        }
        cout << "Here are the acceptable
restaurants:" << endl;
        Set<string> restaurants;
        for (string guest : guests) {
            if (restaurants.isEmpty()) {
                restaurants = map[guest];
            } else {
                restaurants *= map[guest];
            }
        }
        for (string restaurant : restaurants) {
            cout << restaurant << endl;
        }
    }
    return 0;
}
```

Closing Remarks

- Sets/Maps do extend functionality past the vector unlike what we saw with stack/queue. If stack/queue didn't extend functionality, why do we care about them?

- Example counting words in books using a vector and vec.contains(…). Really slow. Now switch vector to set and goes much faster. Why?

- Stack/queue does NOT have for-each loop. That would violate our rule of only being able to see the "next" element.

Look Ahead

- Assignment 1 due **today at 5PM**
- Assignment 2 comes out today, due **Wednesday, July 10 at 5PM**