Programming Abstractions

CS106B

Ashley Taylor

Thanks to Cynthia Lee and Marty Stepp for letting me adapt their slides

Today's Topics

Code Example of File Reading

- countUniqueWords example

Sets (and Lexicons)

- A new kind of ADT
- countUniqueWords redux

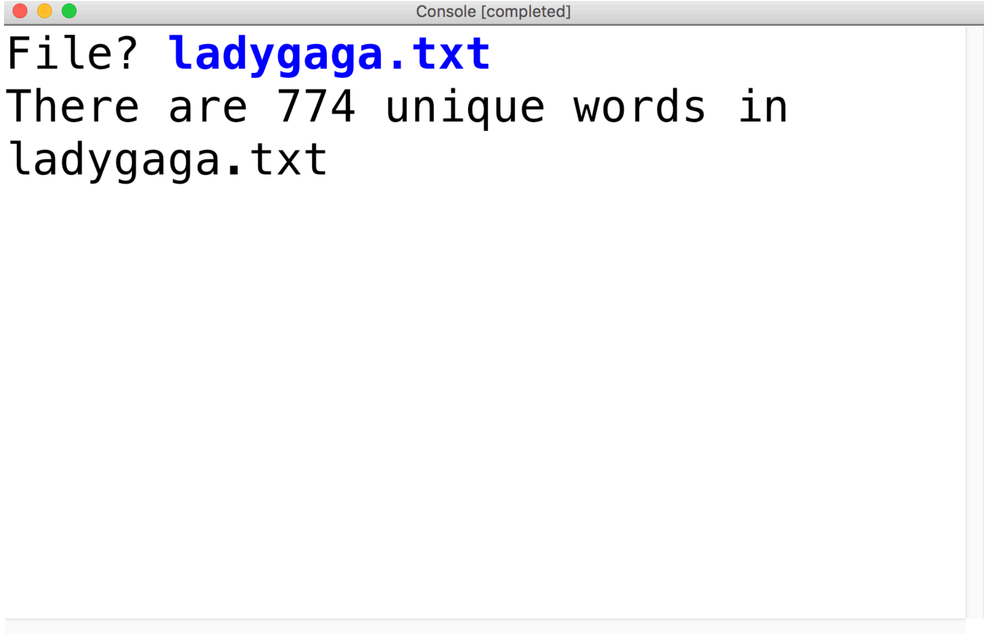
Maps

- An ADT for pairs of data
- wordCount example
- Where2Eat

File Reading Example

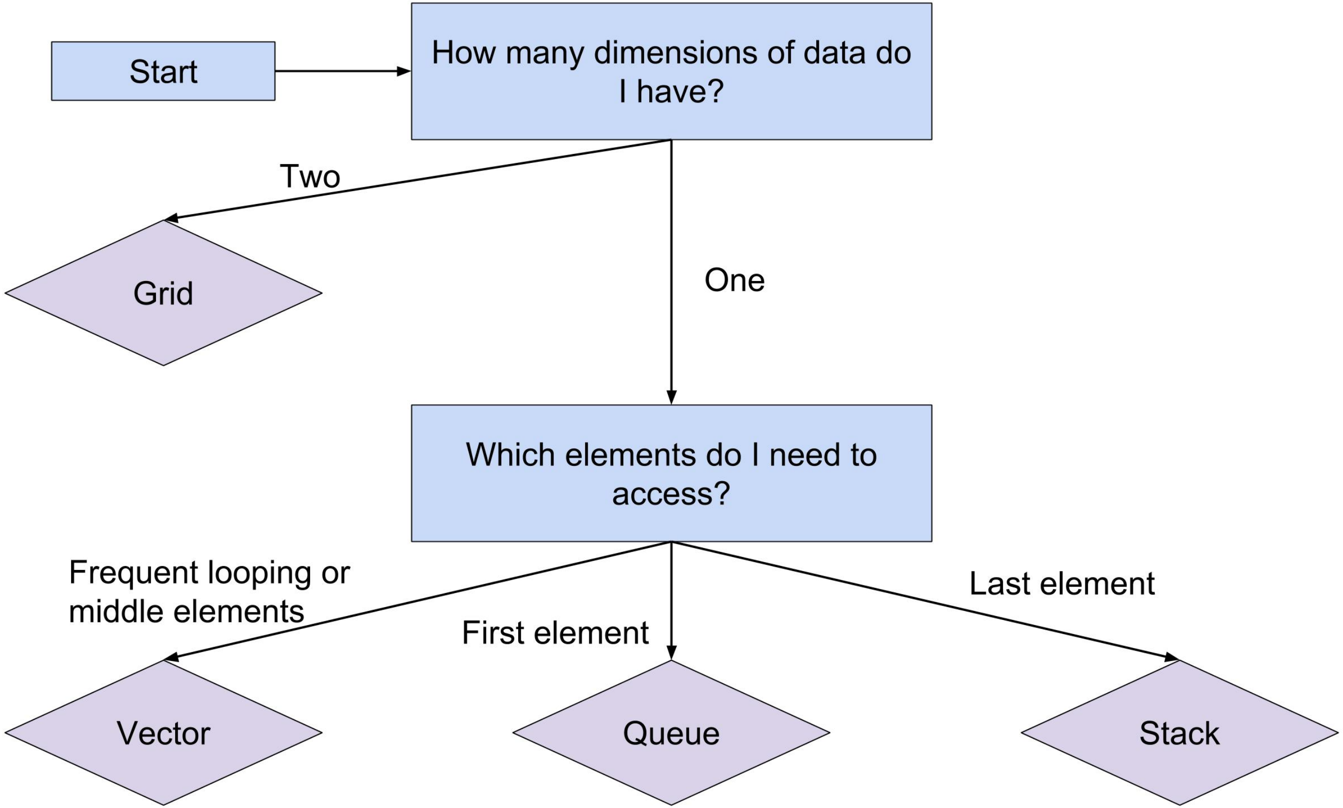
File Reading Example: CountUniqueWords

- 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?



```
File? ladygaga.txt
There are 774 unique words in
ladygaga.txt
```

CountUniqueWords: Which ADT?

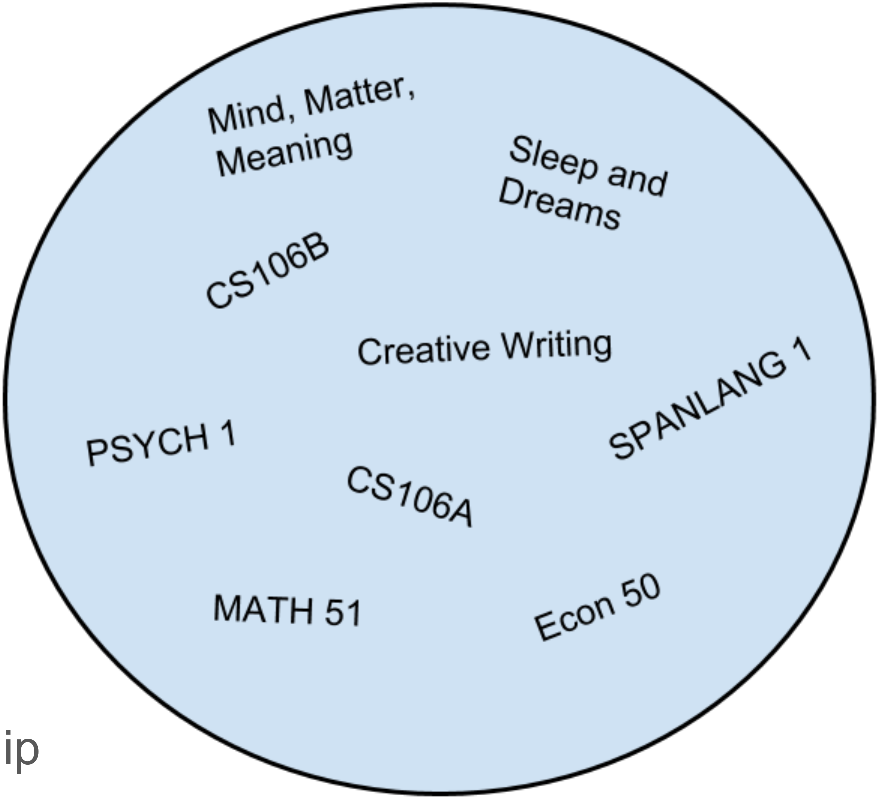

Start
How many dimensions of data do I have?
Two
Grid
One
Which elements do I need to access?
Frequent looping or middle elements
First element
Last element
Vector
Queue
Stack

Sets

An ADT that specializes in membership

Sets

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



Looping over Sets

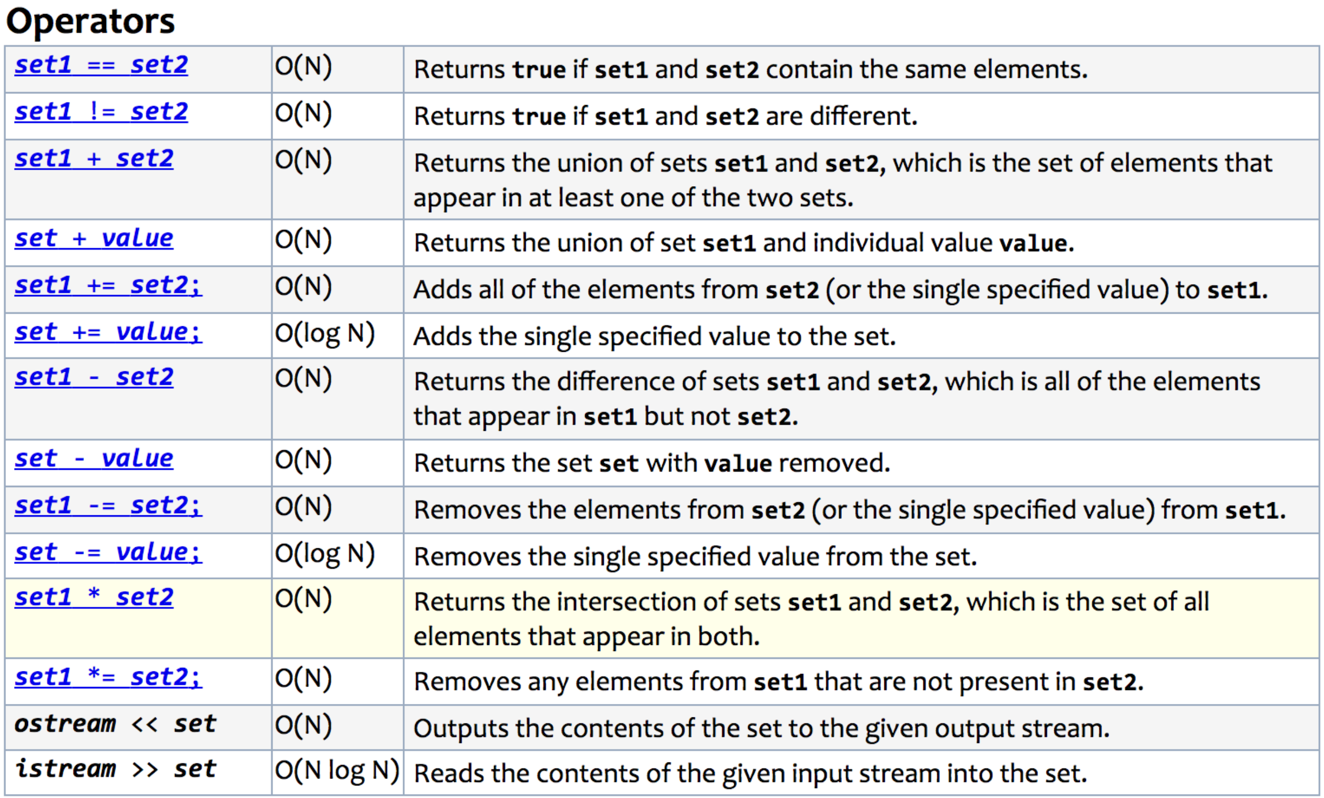
- Sets don’t have indices, so we use a for-each loop
- Iterates in sorted order (alphabetical order for strings)
- Can’t edit while we iterate

```
Set<string> friends;
// add your friends

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

Sets – Other Operations

- Sets are easy to compare and combine



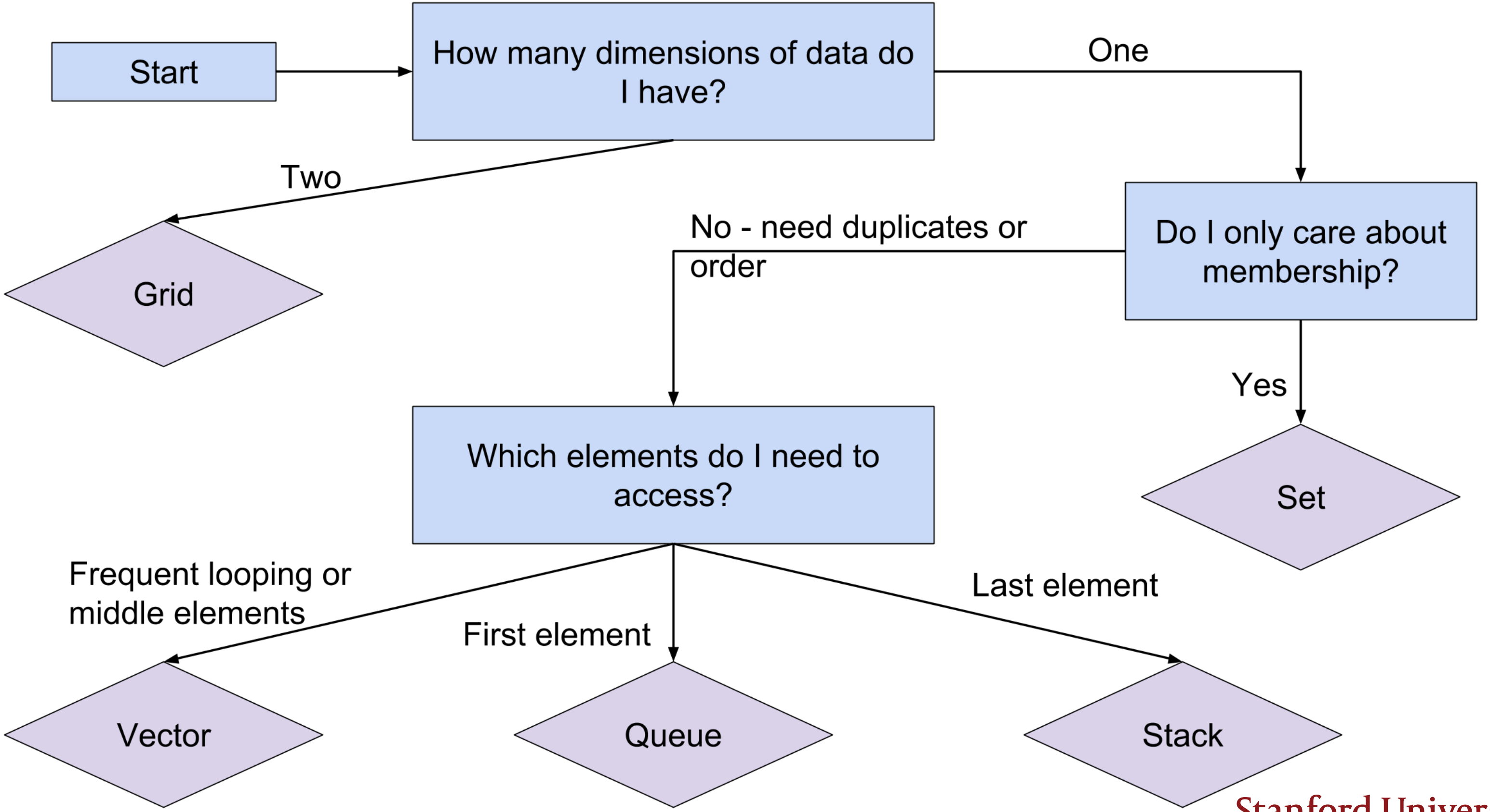
Operators

`set1 == set2`	O(N)	Returns **true** if **set1** and **set2** contain the same elements.
`set1 != set2`	O(N)	Returns **true** if **set1** and **set2** are different.
`set1 + set2`	O(N)	Returns the union of sets **set1** and **set2**, which is the set of elements that appear in at least one of the two sets.
`set + value`	O(N)	Returns the union of set **set1** and individual value **value**.
`set1 += set2;`	O(N)	Adds all of the elements from **set2** (or the single specified value) to **set1**.
`set += value;`	O(log N)	Adds the single specified value to the set.
`set1 - set2`	O(N)	Returns the difference of sets **set1** and **set2**, which is all of the elements that appear in **set1** but not **set2**.
`set - value`	O(N)	Returns the set **set** with **value** removed.
`set1 -= set2;`	O(N)	Removes the elements from **set2** (or the single specified value) from **set1**.
`set -= value;`	O(log N)	Removes the single specified value from the set.
`set1 * set2`	O(N)	Returns the intersection of sets **set1** and **set2**, which is the set of all elements that appear in both.
`set1 *= set2;`	O(N)	Removes any elements from **set1** that are not present in **set2**.
`ostream << set`	O(N)	Outputs the contents of the set to the given output stream.
`istream >> set`	O(N log N)	Reads the contents of the given input stream into the set.

Lexicons – A Special Kind of Set

- 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

ADT Soup Continued



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

Announcements

Assignment 2 YEAH Hours

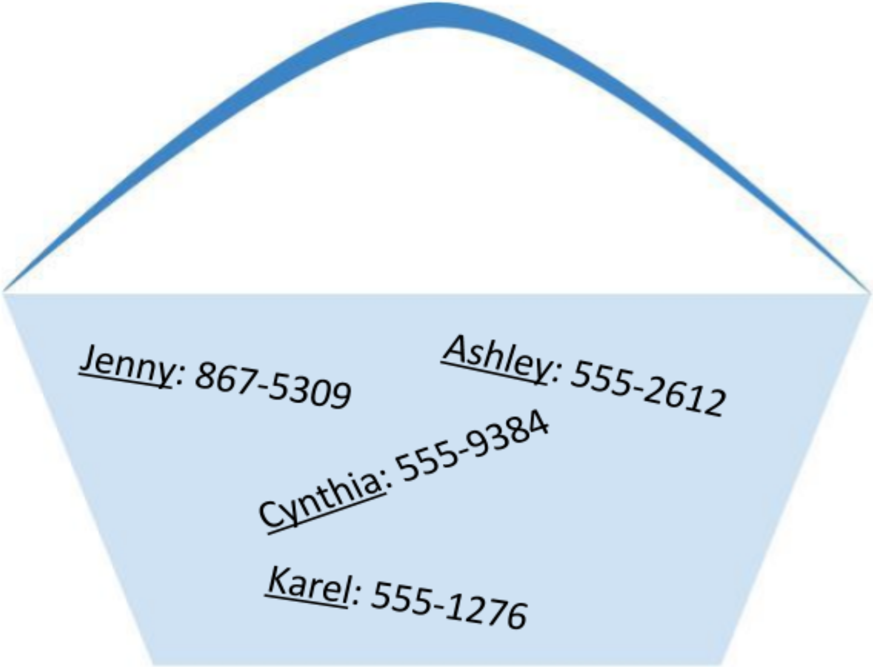
- Monday, 7:00-8:30PM
- Held by awesome SL Nolan
- Location will be announced on Piazza

Maps

A Map is an ADT best at storing **pairs**

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
  - Useful when you will only have half the information available later in the program
  - 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 have 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 (alphabetical order for strings)
- Can’t edit the keys while we iterate (can edit values)

```
Map<string, int> phonebook;
// add your friends to your phonebook

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

Map Example: WordCount

- 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?

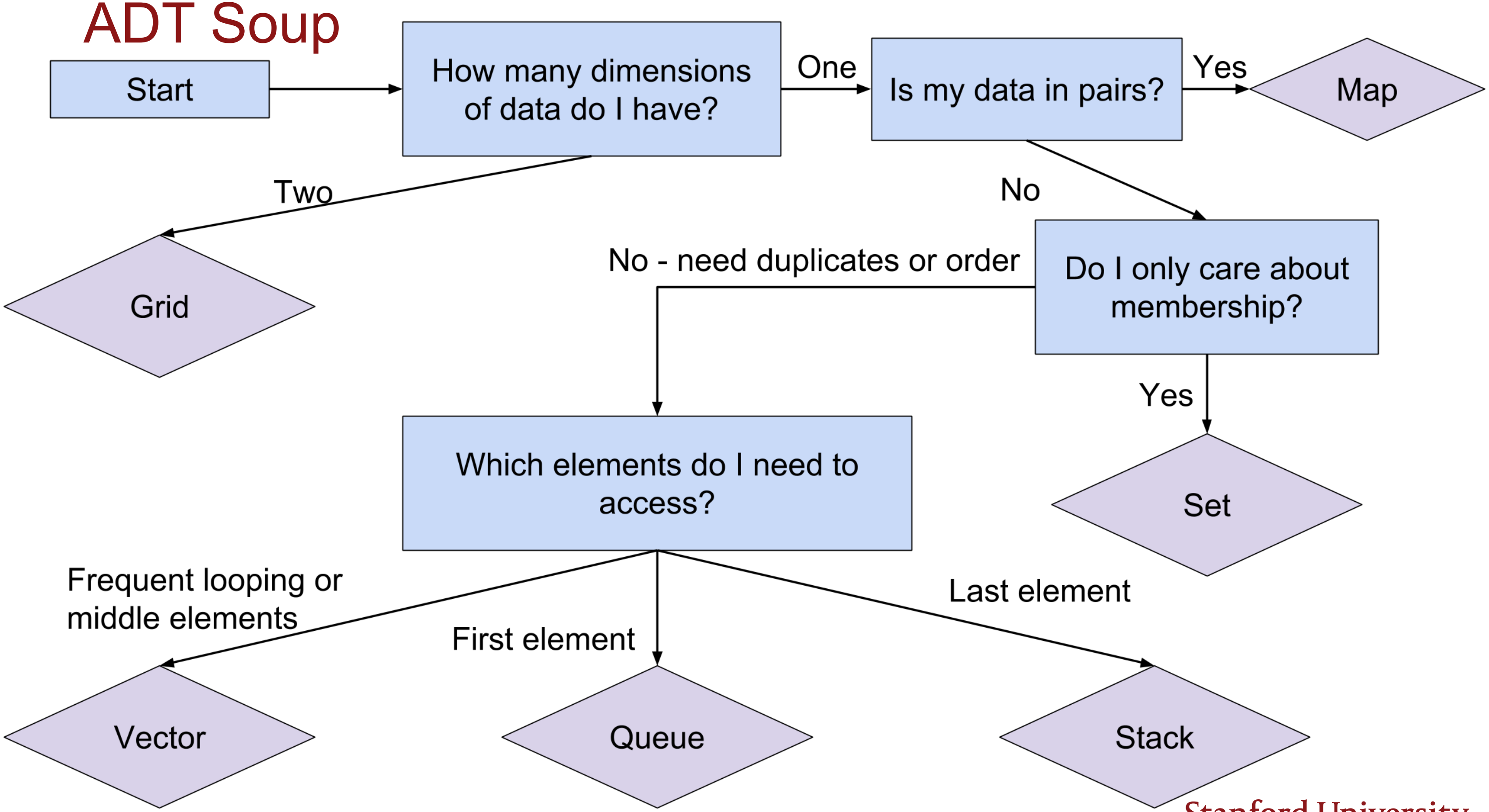
```
to be or not to be
```

```
File? tiny.txt

Word? to
“to” appears 2 times

Word? or
“or” appears 1 times
```

ADT Soup


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

Nesting ADTs: Where2Eat

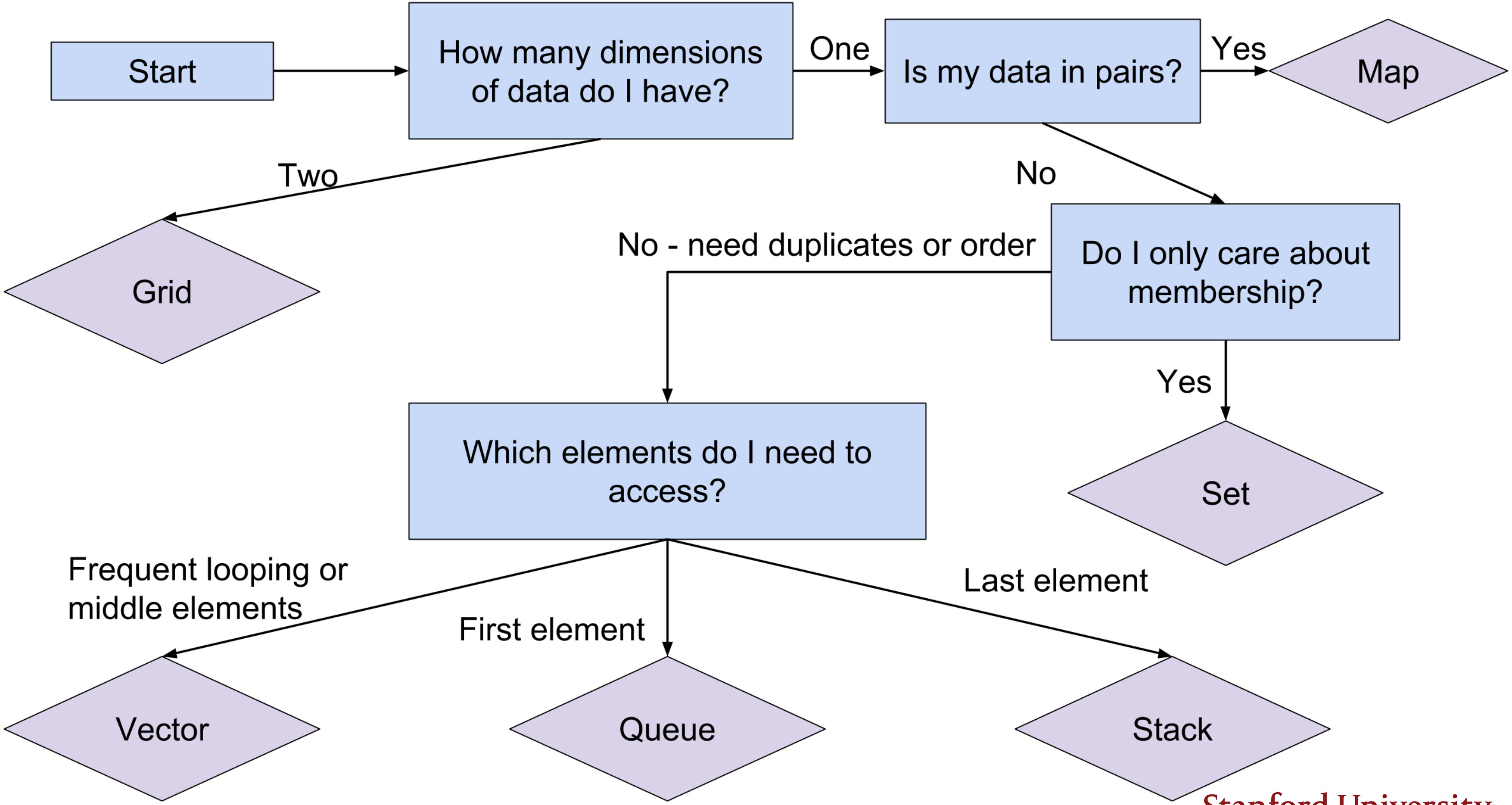
- Problem: we want to schedule a dinner with some group of our friends
- We have a text file with all our friends dinner preferences
- Given a group of friends going to a dinner, what should we eat?

```
Ashley
In n Out
Chipotle
Axe and Palm

Cynthia
Chipotle
Bytes Cafe

Karel
Axe and Palm
Bytes Cafe
```

Which ADTs?



Start

How many dimensions of data do I have?

One

Is my data in pairs?

Yes

Map

Two

Grid

No

Do I only care about membership?

Yes

Set

No - need duplicates or order

Which elements do I need to access?

Frequent looping or middle elements

Vector

First element

Queue

Last element

Stack