Section 5:
HW6 and Interfaces

SLIDES ADAPTED FROM ALEX MARIAKAKIS,
WITH MATERIAL FROM KRYSTA YOUSOUFIAN, MIKE ERNST, KELLEN DONOHUE
How is Homework 5 going?
Agenda

- Reminders
  - HW 5 due tonight (7/19)
  - HW 6 due next Thursday (7/26)

- Breadth-first search (BFS)

- Interfaces

- Parsing Marvel Data
Reminders:

Expensive CheckReps are BAD
(at least when assignments are turned in, but can be useful for finding hard-to-discover problems – so need to be able to control expensive checks)

Debug flags are GOOD
(or enums to indicate depth of debug)
public void checkRep() {
    if (debug) {
        // expensive checks
        BigInteger n = countAtomsInUniverse();
        assert n.equals(theCorrectValue);
        ...
    }
    // cheap checks
    int n = countFingersOnRightHand();
    assert n <= 5 : "beware the six-fingered man";
    ...
}
Don’t forget your CheckReps!
Graphs

Can I reach B from A?
Breadth-First Search (BFS)

- Often used for discovering connectivity
- Calculates the shortest path *if and only if* all edges have same positive or no weight
- Depth-first search (DFS) is commonly mentioned with BFS
  - BFS looks “wide”, DFS looks “deep”
  - DFS can also be used for discovery, but not the shortest path
Breadth-First Search (BFS)

Starting at A, which nodes will be visited first in a BFS?
Breadth-First Search (BFS)

Starting at A, which nodes will be visited first in a BFS? B, C, D
Breadth-First Search (BFS)

Starting at A, which nodes will be visited second in a BFS?
Starting at A, which nodes will be visited second in a BFS? E, F, G
BFS Pseudocode

put start node in a queue
while (queue is not empty):
    pop node N off queue

    if (N is goal):
        return true
    else:
        for each node O that is child of N:
            push O onto queue

return false
Breadth-First Search

START:
Q: <A>
Pop: A, Q: <>
Q: <B, C>
Pop: B, Q: <C>
Q: <C>
Pop: C, Q: <C>
Q: <>
DONE

Starting at A
Goal: Fully explore
Breadth-First Search with Cycle

START:
Q: <A>
Pop: A, Q: <>
Q: <B>
Pop: B, Q: <>
Q: <C>
Pop: C, Q: <>
Q: <A>
NEVER DONE

Starting at A
Goal: Fully Explore
BFS Pseudocode

put start node in a queue
while (queue is not empty):
    pop node $N$ off queue
    mark node $N$ as visited
    if (N is goal):
        return true
    else:
        for each node $O$ that is child of $N$:
            if $O$ is not marked visited:
                push $O$ onto queue
return false
Breadth-First Search

Problem: Find everything reachable from A

Q: <>

Diagram:

- Node A
- Node B
- Node C
- Node D
- Node E

Connections:
- A to B
- A to C
- B to D
- C to D
- D to E
Breadth-First Search

Q: <>
Q: <A>
Breadth-First Search

Q: <>
Q: <A>
Q: <>
Breadth-First Search

Q: <>
Q: <A>
Q: <>
Q: <C>
Breadth-First Search

Q: <>
Q: <A>
Q: <>
Q: <C>
Q: <C,D>
Q: <C,D>
Breadth-First Search

Q: <>
Q: <A>
Q: <>
Q: <C>
Q: <C,D>
Q: <D>
Breadth-First Search

Q: <>
Q: <A>
Q: <>
Q: <C>
Q: <C, D>
Q: <D>
Q: <D, E>
Breadth-First Search

Q: <>
Q: <A>
Q: <>
Q: <C>
Q: <C, D>
Q: <D>
Q: <D, E>
Q: <E>
Breadth-First Search

DONE
Shortest Paths with BFS

**From Node B**

<table>
<thead>
<tr>
<th>Destination</th>
<th>Path</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>&lt;B,A&gt;</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>&lt;B&gt;</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>&lt;B,A,C&gt;</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Shortest path to D? to E? What are the costs?
Shortest Paths with BFS

From Node B

<table>
<thead>
<tr>
<th>Destination</th>
<th>Path</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>&lt;B,A&gt;</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>&lt;B&gt;</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>&lt;B,A,C&gt;</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>&lt;B,D&gt;</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>&lt;B,D,E&gt;</td>
<td>2</td>
</tr>
</tbody>
</table>

Shortest path to D? to E? What are the costs?
Shortest Paths with Weights

Weights are not the same! Are the paths?
Shortest Paths with Weights

From Node B

<table>
<thead>
<tr>
<th>Destination</th>
<th>Path</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>&lt;B,A&gt;</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>&lt;B&gt;</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>&lt;B,A,C&gt;</td>
<td>5</td>
</tr>
<tr>
<td>D</td>
<td>&lt;B,A,C,D&gt;</td>
<td>7</td>
</tr>
<tr>
<td>E</td>
<td>&lt;B,A,C,E&gt;</td>
<td>7</td>
</tr>
</tbody>
</table>
Interfaces
Classes, Interfaces, and Types

- The fundamental unit of programming in Java is a class
- Classes can extend other classes and implement interfaces
- Interfaces can extend other interfaces
Classes, Objects, and Java

Everything is an instance of a class
- Defines data and methods

Every class extends exactly one other class
- Object if no explicit superclass
- Inherits superclass fields

Every class also defines a type
- Foo defines type Foo
- Foo inherits all inherited types
Interfaces

Pure type declaration

```java
public interface Comparable {
    int compareTo(Object other);
}
```

Can contain:
- Method specifications (implicitly `public abstract`)
- Named constants (implicitly `public final static`)

**Does not contain implementation!**

Cannot create instances of interfaces
Implementing Interfaces

- A class can implement one or more interfaces
  ```java
class Kitten implements Pettable, Huggable
  ```
- The implementing class and its instances have the interface type(s) as well as the class type(s)
- The class must provide or inherit an implementation of all methods defined by the interface(s)
  - Not true for abstract classes
Using Interface Types

- An interface defines a type, so we can declare variables and parameters of that type.
- A variable with an interface type can refer to an object of any class implementing that type.

```java
List<String> x = new ArrayList<String>();
void sort(List aList) {...}
```
Guidelines for Interfaces

● Provide interfaces for significant types and abstractions
● Write code using interface types like Map instead of HashMap and TreeMap wherever possible
  ◦ Allows code to work with different implementations later on
● Both interfaces and classes are appropriate in various circumstances
Parsing Marvel Data

- Data is in marvel.tsv
  - Will be pushed with hw6
- Each line is in the form:
  - "character" "book"
  - Ex: "CAPTAIN AMERICA" "N 57"
- Parsing is already implemented for you!
Parsing Marvel Data

- MarvelParser.parseData(String filename, Set<String> characters, Map<String, List<String>> books)
- Call parseData() with an empty Set, Map
- parseData() will fill the Set with all comic book characters, Map with Characters → List of books they’re in
HW 6 Demo