CS2123 Data Structures
Introduction

Weining Zhang
Department of Computer Science
University of Texas at San Antonio
August 23, 2011

Outline

1 Course Orientation
   - Instructors
   - Course Policies

2 Introduction
   - Analysis of Algorithms
   - Universal Modeling Language
   - Exception Handling

Instructor and Presenter

Instructor
Office: SB 4.01.19
Phone: 458-5557
Email: wzhang@cs.utsa.edu
Office Hour: MWF 10:00-11:00am
and by appointment

Lab Presenter
Gregory Hoffer <ghoffer@cs.utsa.edu>

Teaching Support

- Textbook is required
- Instructor’s course web page include lecture notes, case study projects
- Eclipse (www.eclipse.org)
  - interactive development environment (IDE) using Java 6.0
- Blackboard (http://bb.utsa.edu) courseware
  - access to syllabus, calendar, lecture web pages, emails, etc.
  - submit labs and projects, check grades, take quizzes

Blackboard Login
userID: myUTSA ID (“abc123” format)
password: myUTSA password
Course Orientation

Introduction

Course Policies

Grading Policy

- Midterms: 30% (Weeks 7 and 12)
- Final: 30% (Wednesday, Dec. 13, 1:30pm-4:00 pm)
- Quizzes: 10%
  - There are 11 online quizzes, 10 best scores will be counted
- Recitation (CS2121): 15%
  - There are 12 labs. Lab 6 is for career service. The 11 best scores of the labs will be counted. The attendance is required
- Projects: 15%
  - There are 3 projects. They must be individually completed

Other Policies

- Lecture attendance is required, and may affect borderline grades
- No makeup exams
- No late assignments (labs, projects, and quizzes)
- Assignments must be completed Individually

Recitations

- Everyone must register for a recitation section
- Attendance to recitations is required
- Tutors are available to help during designated times in computer labs

Importance of This Course

- CS2123 is a KEY course in our CS curriculum
  - Learn advanced programming techniques
  - Learn common data structures that are fundamental to many other courses
    - analysis of algorithms
    - system programming
    - operating systems
    - programming languages
    - software engineering
    - and more
Definition

An algorithm is a step-by-step instruction that solves a given type of problems.

Example

The following algorithm finds the max integer in an array:
```java
int max = 0;
for each element e in the array do
    if (max < e) max = e;
return max;
```

Growth Functions

Definition

The growth function \( T(n) \) of an algorithm is a formula to compute the number of basic operations performed by the algorithm given the problem size \( n \).

- Count 1 for simple assignment, e.g., \( a = 5 \), or a simple test, e.g., \( x == y \)
- Count \( n \) for following loop
  ```java
  for (int i=0; i<n; i++)
      sum = sum + i;
  ```
- Count \( n^2 \) for the following nested loop
  ```java
  for (int i=0; i<n; i++)
      for (int j=0; j<i; j++)
          System.out.println(i*j + " ");
  ```

Big-O Notation

Definition

The time complexity of an algorithm is \( O(f(n)) \) (read in the order of \( f(n) \)) if its growth function \( T(n) \) is proportional to \( f(n) \).

To find the complexity of an algorithm from its growth function \( T(n) \),
1. find the highest order of \( n \)
2. drop the constant

Example

If the growth function is \( T(n) = 100n + 4n^2 - 26 \), the complexity is \( O(n^2) \).

Compare Algorithms

- Suppose algorithms A and B solves the same types of problems with growth function given in the following table. For what value of \( n \) will A be faster than B?

<table>
<thead>
<tr>
<th>( n )</th>
<th>( T_A(n) = 5n )</th>
<th>( T_B(n) = n^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>6</td>
<td>30</td>
<td>36</td>
</tr>
</tbody>
</table>
Universal Modeling Language

- UML is a graphical language used to describe the design of a software
  - Defines classes, data members (or fields), methods, and relationships among classes
  - Uses standardized graphical representations for various concepts
  - Is popular with software engineers
- UML diagrams are required for all of your programming assignments

UML Basics

- **Class Box**
  - Specify names of class, data fields, methods
  - Specify visibility
    - public: +
    - private: -
- **Class Relationships**
  - Inheritance: super/sub-class (solid-line w/ hollow arrow)
  - Association: many to one, many to many, (solid line w/ number ranges)
  - Aggregation: (solid line w/ hollow diamond)
  - Implement: (dash line w/ hollow arrow)
  - Uses: (dash line w/ simple arrow)
Exception Handling

- Exceptions are objects indicating errors
- Methods can throw an exception if an error occurs
- Calling method use try-catch to catch and to handle an exception

```
try {
    // call some method which may throw an exception
    ....
} catch(Exception e) {
    // handles the exception
    ....
}
```

- Case Study 1: The Currency class throws an
  DataFormatException on incorrect country name