Lecture 01/04/16

Lecturer: Xiaodi Wu

January 4, 2016
Welcome to CIS 313: Intermediate Data Structure
Welcome to CIS 313: Intermediate Data Structure

&

Happy New Year!
Teaching Team

Instructor

- Instructor: Prof. Xiaodi Wu
- Contact: DES 332, xiaodiwu@cs.uoregon.edu
- Research: Quantum Computation
- CIS 410/510 Introduction to Quantum Information Theory, Spring 2016.
Teaching Team

Instructor

- Instructor: Prof. Xiaodi Wu
- Contact: DES 332, xiaodiwu@cs.uoregon.edu
- Research: Quantum Computation
- CIS 410/510 Introduction to Quantum Information Theory, Spring 2016.

GTFs

- Connor George, cgeorge@cs.uoregon.edu, DES 228
  Labs: T 12:00 - 12:50, F 10:00 - 10:50
- Sudhanshu Sane, ssane@cs.uoregon.edu, DES 232
  Labs: T 17:00 - 17:50, R 15:00 - 15:50
Question

What is your motivation to study this course?
Question

What is your motivation to study this course?

- major requirement; fundamental skills for coders; and for coding interviews
Question

What is your motivation to study this course?

▶ major requirement; fundamental skills for coders; and for coding interviews

CIS 313 is about:

▶ Tools and methods to analyze programs (codes): prove its correctness, calculate its performance, ......
Question

What is your motivation to study this course?

▶ major requirement; fundamental skills for coders; and for coding interviews

CIS 313 is about:

▶ Tools and methods to analyze programs (codes): prove its correctness, calculate its performance, ......

▶ in the context of fundamental data structures. Get familiar with data structures as well!
Question

What is your motivation to study this course?

▶ major requirement; fundamental skills for coders; and for coding interviews

CIS 313 is about:

▶ Tools and methods to analyze programs (codes): prove its correctness, calculate its performance, ......
▶ in the context of fundamental data structures. Get familiar with data structures as well!
▶ CIS 315 is in the context of fundamental algorithms.
Information about the course

▶ **Course website**: syllabus, announcements, lecture notes, and so on. Check frequently.
Information about the course

- **Course website**: syllabus, announcements, lecture notes, and so on. Check frequently.
- **Piazza**: discussion forum, announcements, solutions to assignments and exams.
Information about the course

- **Course website**: syllabus, announcements, lecture notes, and so on. Check frequently.
- **Piazza**: discussion forum, announcements, solutions to assignments and exams.
- **Canvas**: distribute and submit lab assignments.

What if I have more questions?
- Check FAQs at the course website.
- Email me (with “CIS 313” in subject) or GTFs.
Information about the course

- **Course website**: syllabus, announcements, lecture notes, and so on. Check frequently.
- **Piazza**: discussion forum, announcements, solutions to assignments and exams.
- **Canvas**: distribute and submit lab assignments.

What if I have more questions?

- Check FAQs at the course website.
- Email me (with “CIS 313” in subject) or GTFs.
Evaluation

- **Active in-class participation (5%)**: reward students asking and answering questions. Attending labs is mandatory.
Evaluation

- **Active in-class participation (5%)**: reward students asking and answering questions. Attending labs is mandatory.

- **Homework and Labs (40%)**: 5 written assignments (10% bonus) and 3 lab assignments. 8 in total, 5% each, work load 1 per week. No assignment due on Week 1 and Week 5 (for midterm).
Evaluation

- **Active in-class participation (5%)**: reward students asking and answering questions. Attending labs is mandatory.

- **Homework and Labs (40%)**: 5 written assignments (10% bonus) and 3 lab assignments. 8 in total, 5% each, work load 1 per week. No assignment due on Week 1 and Week 5 (for midterm).

- **Exams**: midterm (25%) tentative date 02/08/16, **final (30%)** 03/16/16. One review lecture for each exam.
Evaluation

- **Active in-class participation (5%)**: reward students asking and answering questions. Attending labs is mandatory.

- **Homework and Labs (40%)**: 5 written assignments (10% bonus) and 3 lab assignments. 8 in total, 5% each, work load 1 per week. No assignment due on Week 1 and Week 5 (for midterm).

- **Exams**: midterm (25%) tentative date 02/08/16, final (30%) 03/16/16. One review lecture for each exam.

Lectures

- Slides and notes will be posted at the course website after class.
Evaluation

- **Active in-class participation (5%)**: reward students asking and answering questions. Attending labs is mandatory.

- **Homework and Labs (40%)**: 5 written assignments (10% bonus) and 3 lab assignments. 8 in total, 5% each, work load 1 per week. No assignment due on Week 1 and Week 5 (for midterm).

- **Exams: midterm (25%) tentative date 02/08/16, final (30%) 03/16/16.** One review lecture for each exam.

Lectures

- Slides and notes will be posted at the course website after class.

- Reading assignment: different reading requirements. Make good use of the textbook.
Week 1 Arrangement

Office hours

- Wu: M 1:15pm - 2:45pm
- More TBD. available soon!
Week 1 Arrangement

Office hours

- Wu: M 1:15pm - 2:45pm
- More TBD. available soon!

Labs

- No lab. No physical appearance required.
Week 1 Arrangement

Office hours

- Wu: M 1:15pm - 2:45pm
- More TBD. available soon!

Labs

- No lab. No physical appearance required.
- You are expected to get familiar with Java if you haven’t done so. References at the course website (also in email announcements).
Week 1 Arrangement

Office hours

▶ Wu: M 1:15pm - 2:45pm
▶ More TBD. available soon!

Labs

▶ No lab. No physical appearance required.
▶ You are expected to get familiar with Java if you haven’t done so. References at the course website (also in email announcements).
▶ Your lab leader will hold office hours instead, at the lab time and at his office, if you have any questions.
Analysis of programs

Experimental Study: drawbacks

▶ Limited input cases; need good test datas.
▶ Hard to compare efficiency between algorithms due to hardware and software environments.
▶ One needs to do the coding!

Ideal Analytical Framework

▶ Take into account all possible cases.
▶ Compare efficiency between algorithms independent of hardware and software environments.
▶ Need only high-level descriptions.
Analysis of programs

Experimental Study: drawbacks

- Limited input cases; need good test data.

- Hard to compare efficiency between algorithms due to hardware and software environments.

- One needs to do the coding!

Ideal Analytical Framework

- Take into account all possible cases.

- Compare efficiency between algorithms independent of hardware and software environments.

- Need only high-level descriptions.
Analysis of programs

Experimental Study: drawbacks

- Limited input cases; need good test datas.
- Hard to compare efficiency between algorithms due to hardware and software environments.
Analysis of programs

Experimental Study: drawbacks

- Limited input cases; need good test data.
- Hard to compare efficiency between algorithms due to hardware and software environments.
- One needs to do the coding!
Analysis of programs

Experimental Study: drawbacks

▶ Limited input cases; need good test datas.
▶ Hard to compare efficiency between algorithms due to hardware and software environments.
▶ One needs to do the coding!

Ideal Analytical Framework

▶ Take into account all possible cases.
Analysis of programs

Experimental Study: drawbacks

- Limited input cases; need good test data.
- Hard to compare efficiency between algorithms due to hardware and software environments.
- One needs to do the coding!

Ideal Analytical Framework

- Take into account all possible cases.
- Compare efficiency between algorithms independent of hardware and software environments.
Analysis of programs

Experimental Study: drawbacks

▶ Limited input cases; need good test data.
▶ Hard to compare efficiency between algorithms due to hardware and software environments.
▶ One needs to do the coding!

Ideal Analytical Framework

▶ Take into account all possible cases.
▶ Compare efficiency between algorithms independent of hardware and software environments.
▶ Need only high-level descriptions.
High-level descriptions: pseudo-codes

Example: algorithm 1.2 in Chapter 1.1.1

Algorithm: arrayMax(A,n):

Input : An array A storing $n \geq 1$ integers.
Output : The maximum elements in A.

$\textit{currentMax} \leftarrow A[0]$
for $i \leftarrow 1$ to $n - 1$ do
    if $\textit{currentMax} < A[i]$ then
        $\textit{currentMax} \leftarrow A[i]$
    end if
end for
return $\textit{currentMax}$
Comparisons: modeling of computation costs

Primitive Operations: Unit 1

- Assigning a value to a variable; Indexing into an array; Comparing two numbers;
- An arithmetic operation (+); Calling and returning from a method; ......
Comparisons: modeling of computation costs

Primitive Operations: Unit 1

- Assigning a value to a variable; Indexing into an array; Comparing two numbers;
- An arithmetic operation (+); Calling and returning from a method; ......

Pros and Cons

Pros: only need to count the number of primitive operations.
Comparisons: modeling of computation costs

Primitive Operations: Unit 1

- Assigning a value to a variable; Indexing into an array; Comparing two numbers;
- An arithmetic operation (+); Calling and returning from a method; ......

Pros and Cons

Pros: only need to count the number of primitive operations.

Cons: model-specific (input encoding, OS, ...) ; exact number is too labor-intense and unnecessary.
Comparisons: modeling of computation costs

Primitive Operations: Unit 1

- Assigning a value to a variable; Indexing into an array; Comparing two numbers;
- An arithmetic operation (+); Calling and returning from a method; ......

Pros and Cons

Pros: only need to count the number of primitive operations.

Cons: model-specific (input encoding, OS, ...); exact number is too labor-intense and unnecessary.

Caution: (×) a primitive operation?
Worst case vs Average Case

Worst case analysis: adopted in this course

- Maximum number of primitive operations over all cases.
- Hardest case! Usually easier to analyze than average case study.
Worst case vs Average Case

Worst case analysis: adopted in this course

- Maximum number of primitive operations over all cases.
- Hardest case! Usually easier to analyze than average case study.
- Worst cases might be rare in practice.
Worst case vs Average Case

Worst case analysis: adopted in this course

- Maximum number of primitive operations over all cases.
- Hardest case! Usually easier to analyze than average case study.
- Worst cases might be rare in practice.

Average case analysis

- Average number of primitive operations over a distribution of all cases.
- Average performance. Harder to analyze. Quality depends on the distribution.
Worst case vs Average Case

Worst case analysis: adopted in this course

- Maximum number of primitive operations over all cases.
- Hardest case! Usually easier to analyze than average case study.
- Worst cases might be rare in practice.

Average case analysis

- Average number of primitive operations over a distribution of all cases.
- Average performance. Harder to analyze. Quality depends on the distribution.
Asymptotic Notation

Goals

- Insensitive to constant factors: (1) different models generate constant factors (2) no capture of asymptotic behavior.

Definition (Big-O)

Given \( f, g: \mathbb{N} \rightarrow \mathbb{R}_+ \), \( f(n) \) is \( O(g(n)) \) if \( \exists \) constant \( c > 0 \) and \( n_0 \in \mathbb{N} \) such that \( f(n) \leq cg(n) \), \( \forall n \geq n_0 \).

\( f(n) \) is big-Oh of \( g(n) \) (or, order \( g(n) \)).
Asymptotic Notation

Goals

▶ Insensitive to constant factors: (1) different models generate constant factors (2) no capture of asymptotic behavior.
▶ Capture dominating terms and simplify analysis.
Asymptotic Notation

Goals

- Insensitive to constant factors: (1) different models generate constant factors (2) no capture of asymptotic behavior.
- Capture dominating terms and simplify analysis.
- Demonstrate asymptotic behavior. A guideline for determining efficiency (See Chapter 1.2.3).

Definition (Big-O)

Given $f, g: \mathbb{N} \rightarrow \mathbb{R}^+$, $f(n)$ is $O(g(n))$ if $\exists$ constant $c > 0$ and $n_0 \in \mathbb{N}$ such that $f(n) \leq cg(n)$, $\forall n \geq n_0$.

$f(n)$ is big-Oh of $g(n)$ (or, order $g(n)$).
Asymptotic Notation

Goals

▶ Insensitive to constant factors: (1) different models generate constant factors (2) no capture of asymptotic behavior.
▶ Capture dominating terms and simplify analysis.
▶ Demonstrate asymptotic behavior. A guideline for determining efficiency (See Chapter 1.2.3).
▶ **Cons**: huge constants impact practical cases (non-asymptotic cases)

---

**Definition (Big-O)**

Given $f, g : \mathbb{N} \to \mathbb{R}^+$, $f(n)$ is $O(g(n))$ if there exists a constant $c > 0$ and $n_0 \in \mathbb{N}$ such that $f(n) \leq cg(n), \forall n \geq n_0$. $f(n)$ is big-Oh of $g(n)$ (or, order $g(n)$).
Asymptotic Notation

Goals

▶ Insensitive to constant factors: (1) different models generate constant factors (2) no capture of asymptotic behavior.
▶ Capture dominating terms and simplify analysis.
▶ Demonstrate asymptotic behavior. A guideline for determining efficiency (See Chapter 1.2.3).
▶ **Cons**: huge constants impact practical cases (non-asymptotic cases)

Definition (Big-O)

Given $f, g : \mathbb{N} \rightarrow \mathbb{R}^+$, $f(n)$ is $O(g(n))$ if $\exists$ constant $c > 0$ and $n_0 \in \mathbb{N}$ such that

$$f(n) \leq cg(n), \forall n \geq n_0.$$ 

$f(n)$ is big-Oh of $g(n)$ (or, order $g(n)$).
Big-O

Examples

- $20n^3 + 10n \log n + 5$ is $O(n^3)$? $O(n^2)$? $O(n^4)$?
- $3 \log n + \log \log n$ is $O(\log n)$? $O(n)$? $O(n^{0.000001})$?
- $2^{2^{100}}$ is $O(1)$? $2^{2^{100}}n$ is $O(n)$?
- $5/n$ is $O(1/n)$? $O(1/\sqrt{n})$? $O(1/n^{1.5})$? $O(1)$?
- By definition and/or by rules in Theorem 1.7.

Common Orders

<table>
<thead>
<tr>
<th>logarithmic</th>
<th>linear</th>
<th>quadratic</th>
<th>polynomial</th>
<th>exponential</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(\log n)$</td>
<td>$O(n)$</td>
<td>$O(n^2)$</td>
<td>$O(n^k)(k \geq 1)$</td>
<td>$O(a^n)(a \geq 1)$</td>
</tr>
</tbody>
</table>