Lecture 09/26/16

Lecturer: Xiaodi Wu

Reading: Course Website, Chapter 1.1, 1.2
Welcome to CIS 313: Intermediate Data Structure
Welcome to CIS 313: Intermediate Data Structure
&
Welcome to a new academic year!
Teaching Team

Instructor

- Instructor: Prof. Xiaodi Wu
- Contact: DES 332, xiaodiwu@cs.uoregon.edu
- Research: Quantum Computation
Teaching Team

Instructor

- Instructor: Prof. Xiaodi Wu
- Contact: DES 332, xiaodiwu@cs.uoregon.edu
- Research: Quantum Computation

GTFs

- Raleigh Foster, foster@cs.uoregon.edu, DES 100
  Labs: M 13:00 - 13:50, T 12:00 - 12:50
- Samuel Mergendahl, smergend@cs.uoregon.edu, DES 100
  Labs: R 09:00 - 09:50
More logistics

Office Hours: 11 hrs/week, ≥2 hrs/day

- Wu: M 12:00pm -1:30pm, W 12:00pm-1:30pm, DES 332
- Foster: M 2:00pm- 4:00pm, R 3:00pm-5:00pm, DES 100
- Mergendahl: T 9:00-11:00am, F 1:00-3:00pm, DES 100

Websites

- Course website: syllabus, reading assignments, lecture notes (posted before and modified after lectures), and so on. Check Frequently!!
- Piazza: announcements, discussion forum, ask for helps.
- Canvas: distribute and submit Lab assignments, grades, solutions.
More logistics

Office Hours: 11 hrs/week, ≥2 hrs/day

- Wu: M 12:00pm -1:30pm, W 12:00pm-1:30pm, DES 332
- Foster: M 2:00pm- 4:00pm, R 3:00pm-5:00pm, DES 100
- Mergendahl: T 9:00-11:00am, F 1:00-3:00pm, DES 100

Websites

- **Course website**: syllabus, reading assignments, lecture notes (posted before and modified after lectures), and so on. Check **Frequently!!**.
More logistics

Office Hours: 11 hrs/week, ≥2 hrs/day

- Wu: M 12:00pm -1:30pm, W 12:00pm-1:30pm, DES 332
- Foster: M 2:00pm- 4:00pm, R 3:00pm-5:00pm, DES 100
- Mergendahl: T 9:00-11:00am, F 1:00-3:00pm, DES 100

Websites

- **Course website**: syllabus, reading assignments, lecture notes (posted before and modified after lectures), and so on. Check **Frequently!!**.
- **Piazza**: announcements, discussion forum, ask for helps.
More logistics

Office Hours: 11 hrs/week, ≥2 hrs/day

- Wu: M 12:00pm -1:30pm, W 12:00pm-1:30pm, DES 332
- Foster: M 2:00pm- 4:00pm, R 3:00pm-5:00pm, DES 100
- Mergendahl: T 9:00-11:00am, F 1:00-3:00pm, DES 100

Websites

- **Course website**: syllabus, reading assignments, lecture notes (posted before and modified after lectures), and so on. Check Frequently!!.
- **Piazza**: announcements, discussion forum, ask for helps.
- **Canvas**: distribute and submit Lab assignments, grades, solutions.
Evaluation

- **Active participation (3%)** in lectures, labs, and office hours in strongly encouraged and rewarded. **Bonus: (3%)** for actively helping other students (make us know).
Evaluation

- **Active participation (3%)** in lectures, labs, and office hours in strongly encouraged and rewarded. **Bonus:** (3%) for actively helping other students (make us know).

- **Homework and Labs (42%)**: 4 written assignments and 5 lab assignments. 5% each except Lab 0 (2%). **Bonus:** (17%) for bonus problems in both assignments and labs.

- **Exams**: midterm (25%) tentative date 10/26/16, **final** (30%) 12/08/16. One review lecture for each exam.
Evaluation

▶ **Active participation (3%)** in lectures, labs, and office hours in strongly encouraged and rewarded. **Bonus:** (3%) for actively helping other students (make us know).

▶ **Homework and Labs (42%)**: 4 written assignments and 5 lab assignments. 5% each except Lab 0 (2%). **Bonus:** (17%) for bonus problems in both assignments and labs.

▶ **Exams: midterm (25%)** tentative date 10/26/16, **final (30%)** 12/08/16. One review lecture for each exam.

▶ **Grading**: B- or above if your grade is above the average/median of the class. Final score up to curving.
Labs

What are they about?

- Implementing (Java) data structures from the class!
Labs

What are they about?

- Implementing (Java) data structures from the class!

Schedule

- Lab 0 (one-week lab in Week 1) setting up the format, due 10/02/16.
Labs

What are they about?

- Implementing (Java) data structures from the class!

Schedule

- Lab 0 (one-week lab in Week 1) setting up the format, due 10/02/16.
- Lab 1, 2, 3, 4 (two-week labs). 1st week: description of the lab; 2nd week: help with the lab and the assignment.
Labs

What are they about?

▶ Implementing (Java) data structures from the class!

Schedule

▶ Lab 0 (one-week lab in Week 1) setting up the format, due 10/02/16.
▶ Lab 1, 2, 3, 4 (two-week labs). 1st week: description of the lab; 2nd week: help with the lab and the assignment.
▶ No lab in Week 10. Replaced by office hours for the final.
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 (Google; Facebook; Microsoft; ...)
Question

What is your motivation to study this course?

▶ major requirement; fundamental skills for coders; and for coding interviews (Google; Facebook; Microsoft; ...)

CIS 313 is about:

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

What is your motivation to study this course?

- major requirement; fundamental skills for coders; and for coding interviews (Google; Facebook; Microsoft; ...)

CIS 313 is about:

- Tools and methods to analyze programs (codes): prove its correctness, determine 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 (Google; Facebook; Microsoft; ...)

CIS 313 is about:

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

What is your motivation to study this course?

- major requirement; fundamental skills for coders; and for coding interviews (Google; Facebook; Microsoft; ...)

CIS 313 is about:

- Tools and methods to analyze programs (codes): prove its correctness, determine 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.
- **Hard course:** math maturity (proofs) and coding skills !!.
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 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 data.
- 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 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.
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 $A$.

$$currentMax \leftarrow A[0]$$

**for** $i \leftarrow 1$ **to** $n - 1$ **do**

**if** $currentMax < A[i]$ **then**

$$currentMax \leftarrow A[i]$$

**end if**

**end for**

**return** $currentMax$
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.

Definition (Big-O)

Given $f, g: \mathbb{N} \to \mathbb{R}^+$, $f(n)$ 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).

**Definition (Big-O)**

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