

# Course Introduction

Dr. 何明昕, He Mingxin, Max

program06 @ yeah.net

Email Subject: CS203B + *ID* + *Name: TOPIC*

Sakai: CS203B Fall 2022

数据结构与算法分析B

Data Structures and Algorithm Analysis

# Course Structure

- ▶ **Lectures:** 1-16周, 星期二晚上, 9-10节, 一教107
- ▶ **Labs:** 1-16周

**Lecturer:** 何明昕, HE Mingxin, Max  
[program06@yeah.net](mailto:program06@yeah.net)

**Lab Tutor:** 王大兴(WANG Daxing)老师  
[wangdx3@mail.sustech.edu.cn](mailto:wangdx3@mail.sustech.edu.cn)



CS203B-f22-课程群



该二维码7天内(9月10日前)有效，重新进入将更新

# Course Information

Course Site in Sakai: **CS203B Fall 2022**

Please Search the Site and Add it to your Working Space!

BookSite for Text A: S&W: **Algorithms, 4th Ed.**

<https://algs4.cs.princeton.edu/>

Pre-requisites:

**CS102A/CS102B** (Introduction to Computer Programming A or B)

# Course Objective :

In this course, we will study the **fundamental data structures**, and **basic algorithms analysis methods**.

Topics to be covered include **array, linked list, queue, stack, searching in ordered lists, sorting, priority queues, binary search trees, and fundamental graph algorithms**.

In addition, the algorithm analysis (i.e., **time complexity, space complexity**) methods also will be introduced in this course.

Such knowledge will help non-computer science students to solve problems by computers in their future study.

# Expected Outputs :

On completion of the course, students will have acquired the following knowledge and skills:

- ▶ Be familiar with the fundamentals of data structures.
- ▶ Be familiar with basic algorithms analysis methods.
- ▶ Be able to select appropriate data structures for solving simple computing problems.
- ▶ Be able to design simple algorithms with appropriate data structures and analyse their performance.

# 3 Dimensions of Typical CS Courses

Most of Courses on CS may be organized by a combination of elements from the following 3 dimensions:

**THEORY related:** Concepts, Models, Maths, Algorithms, Principles, Mechanisms, Methods, ... Need to understand **Abstract Things**

**SYSTEM and TOOLS related:** HW, Network, OS, PLs, IDEs, DBMS, Clients, Servers, Virtual Machines, Containers in Cloud, ... Need to **understand, setup and use them properly**

**DESIGN related:** According to Requirements (Problems), need to **use Theory and Tools to shape/create/implement/test Solutions!**

**Put Them All Together!**

# Solve problems by **elegant programming** (The way of the program)

Think like a computer scientist who combines some of the best features of:

- **Mathematicians**: use formal languages to denote ideas
- **Engineers**: design things, assembling components into systems and evaluating tradeoffs among alternatives
- **Scientists**: observe the behavior of complex systems, form hypotheses, and test predictions.



# Elegant Programming 简明/优雅地编程

elegant

英: ['elɪɡənt]   美: ['elɪɡənt]  

adj. 优美的; 文雅的; 漂亮雅致的; 陈设讲究的; 精美的; 简练的; 简洁的; 简明的;

← 从后往前读

**craft** ☆

英 [kra:ft]  美 [kræft] 

n. 工艺; 手艺; 太空船

vt. 精巧地制作

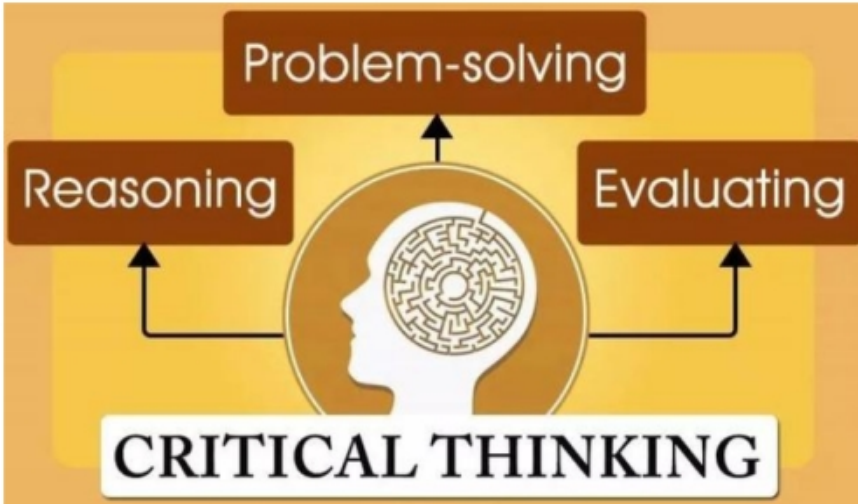
n. (Craft)人名; (英、德、罗)克拉夫特

复数 crafts; 过去式 crafted; 过去分词 crafted; 现在分词 crafting; 第三人称单数 crafts

精心设计制作

# Problem-Solving

- ▶ The single most important skill for a computer scientist is problem-solving.
- ▶ What is **problem-solving**?
  - the ability to formulate problems, think creatively about solutions, and express a solution clearly and accurately
- ▶ The process of **learning to program is an excellent opportunity to practice problem-solving** skills.



- ▶ 批判性思维  
关键性思考  
建设性思考
- ▶ Critical Observing:  
建设性观察
- ▶ 系统化认知问题 与  
构建解决方案的能力



原书第3版

**CRITICAL THINKING**  
TOOLS FOR TAKING CHARGE OF YOUR LEARNING AND YOUR LIFE, 3RD EDITION

**批判性思维工具**

[美] 理查德·保罗 (Richard Paul) | 琳达·埃尔德 (Linda Elder) 著 侯玉波 姜佟琳 等译

风靡美国50年的思维方法

美国“批判性思维国家高层理事会”主席、国际公认的批判性思维  
权威大师 保罗力作

耶鲁、牛津、斯坦福等世界名校最重视的人才培养目标



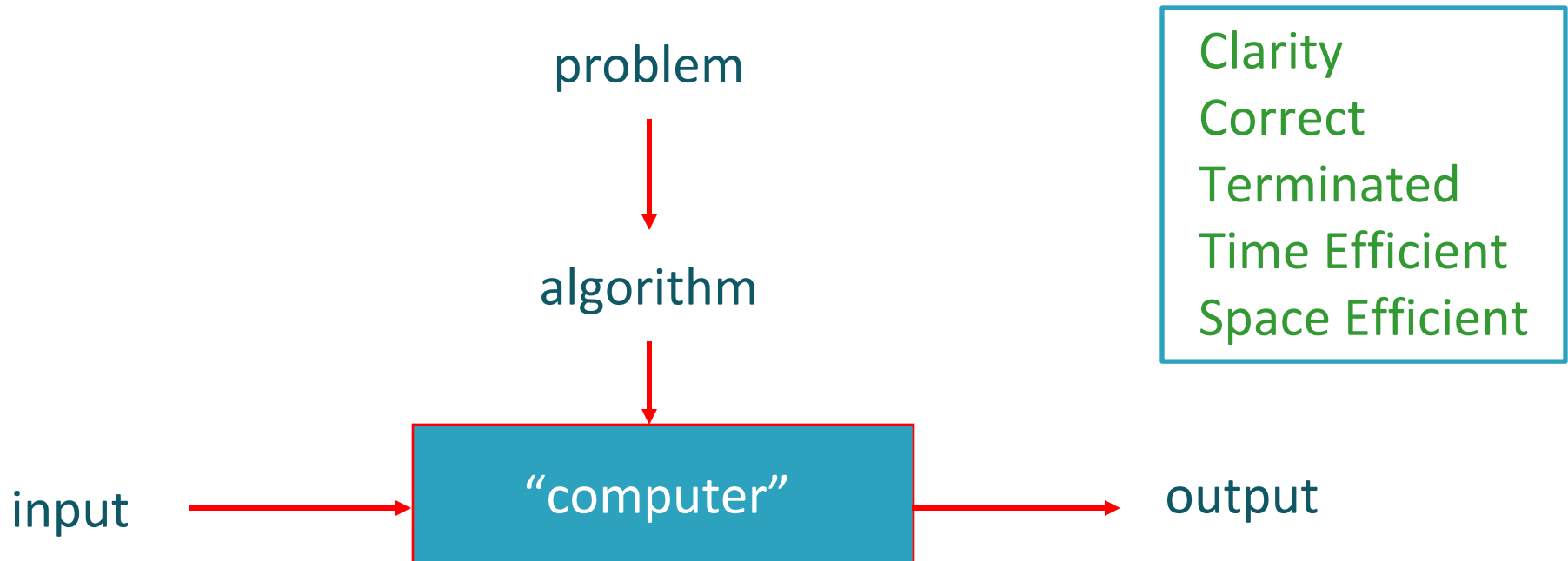
机械工业出版社  
China Machine Press

# Overview

- ▶ Algorithm
  - A **problem-solving method** suitable for implementation as a computer program
- ▶ Data structures
  - Objects created to organize data used in computation
  - **A way to store and organize data in order to facilitate access and modifications**
- ▶ Data structure exist as the by-product or end product of algorithms
  - Understanding data structure is essential to understanding algorithms and hence to problem-solving
  - **Simple algorithms can give rise to complicated data-structures**
  - **Complicated algorithms can use simple data structures**

# What is an Algorithm?

An *algorithm* is a sequence of unambiguous instructions for solving a computation problem, i.e., for obtaining a required output for any legitimate input in a finite amount of time.



# Why study Data Structures and algorithms?

- ▶ Using a computer?
  - Solve computational problems?
  - Want it to go faster?
  - Ability to process more data?
- ▶ Technology vs. Performance/cost factor
  - Technology can improve things by a constant factor
  - Good algorithm design can do much better and may be cheaper
  - Supercomputer cannot rescue a bad algorithm
- ▶ Data structures and algorithms as a field of study
  - Old enough to have basics known
  - New discoveries
  - Burgeoning application areas
  - Philosophical implications?

# Data Structures covered in the course

Data Structures	Advantages	Disadvantages
Array	Quick insertion, very fast access if index known	Slow search (fast after sorting), slow deletion, fixed size.
Ordered array	Quicker search than unsorted array.	Slow insertion and deletion, fixed size.
Stack	Provides last-in, first-out access.	Slow access to other items.
Queue	Provides first-in, first-out access.	Slow access to other items.
Linked list	Quick insertion, quick deletion.	Slow search.
Binary tree	Quick search, insertion, deletion	Deletion algorithm is complex.
Graph	Models real-world situations.	Some algorithms are slow and complex.
Hashing	Quick search, insertion, and deletion	Collision

# Main Topics Covered

(Negotiable, to be tuned)

15 UnionFind

14 AnalysisOfAlgorithms

13 StacksAndQueues

21 ElementarySorts

22 Mergesort

23 Quicksort

24 PriorityQueues

31 ElementarySymbolTables

32 BinarySearchTrees

33 BalancedSearchTrees

41 UndirectedGraphs

42 DirectedGraphs

43 MinimumSpanningTrees

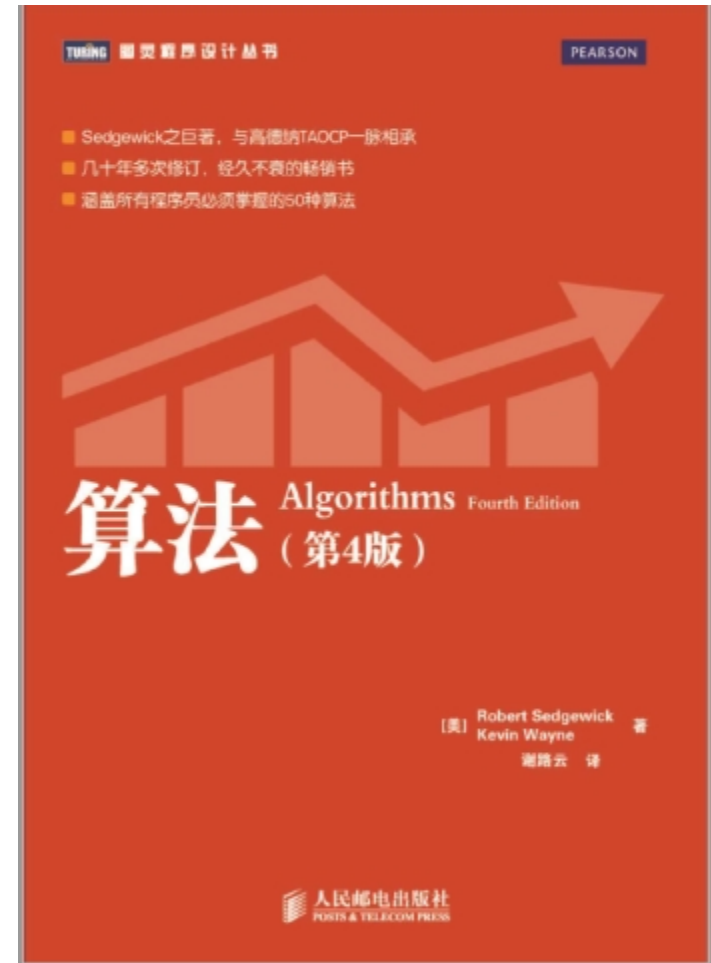
51 StringSorts

53 SubstringSearch



# TextBook A

□ Booksite (Thanks Robert Sedgewick and Kevin Wayne from Princeton Univ. ):  
<http://algs4.cs.princeton.edu/>




# <http://algs4.cs.princeton.edu/>

## ALGORITHMS, 4TH EDITION

*essential information that  
every serious programmer  
needs to know about  
algorithms and data structures*

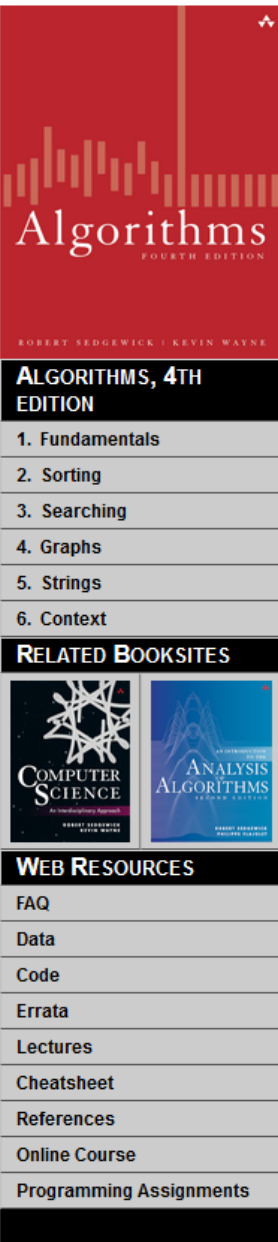
**Online content.** This booksite contains tens of thousands of files, fully coordinated with our textbook and also useful as a stand-alone resource. It consists of the following elements:

- *Excerpts.* A condensed version of the text narrative, for reference while online.
- *Lectures.* Curated studio-produced online videos, suitable for remote instruction via [CUvids](#) .
- *Java code.* The algorithms and clients in this textbook, along with the standard libraries they use.
- *Exercises.* Selected exercises from the book and “web exercises” developed since its publication, along with solutions to selected exercises.
- *Programming assignments.* Creative programming assignments that we have used at Princeton.

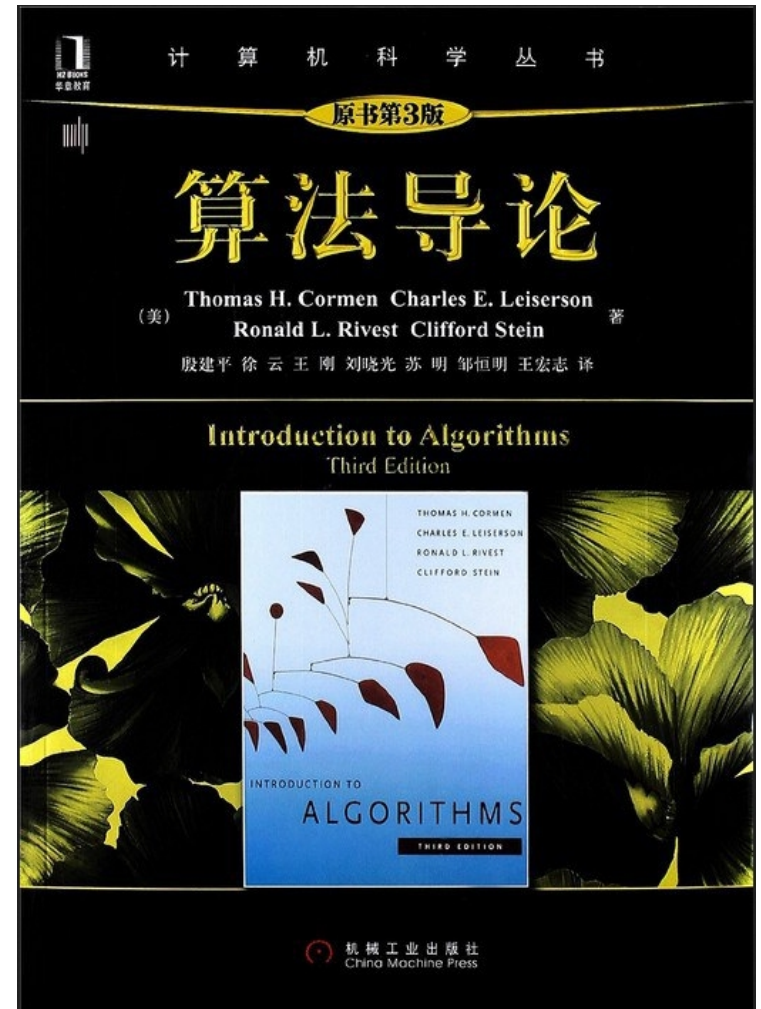
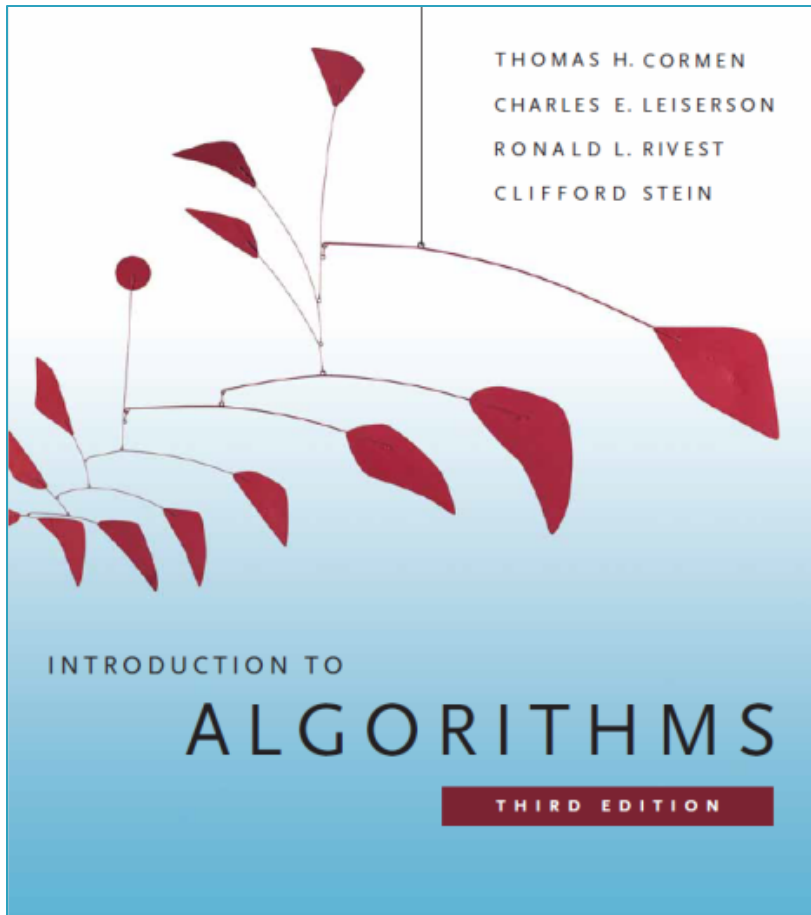
You can explore these resources via the sidebar at left.

**Textbook.** The textbook *Algorithms, 4th Edition* by Robert Sedgewick and Kevin Wayne [ [Amazon](#) · [Pearson](#) · [InformIT](#) ] surveys the most important algorithms and data structures in use today. We motivate each algorithm that we address by examining its impact on applications to science, engineering, and industry. The textbook is organized into six chapters:

- [Chapter 1: Fundamentals](#) introduces a scientific and engineering basis for comparing algorithms and making predictions. It also includes our programming model.
- [Chapter 2: Sorting](#) considers several classic sorting algorithms, including insertion sort, mergesort, and quicksort. It also features a binary heap implementation of a priority queue.
- [Chapter 3: Searching](#) describes several classic symbol-table implementations, including binary search trees, red–black trees, and hash tables.
- [Chapter 4: Graphs](#) surveys the most important graph-processing problems, including depth-first search, breadth-first search, minimum spanning trees, and shortest paths.
- [Chapter 5: Strings](#) investigates specialized algorithms for string processing, including radix sorting, substring search, tries, regular expressions, and data compression.
- [Chapter 6: Context](#) highlights connections to systems programming, scientific computing, commercial applications, operations research, and intractability.



# TextBook B



# Reference A: Good Open Book for Java Programming

<http://introcs.cs.princeton.edu/java/>

## COMPUTER SCIENCE: AN INTERDISCIPLINARY APPROACH

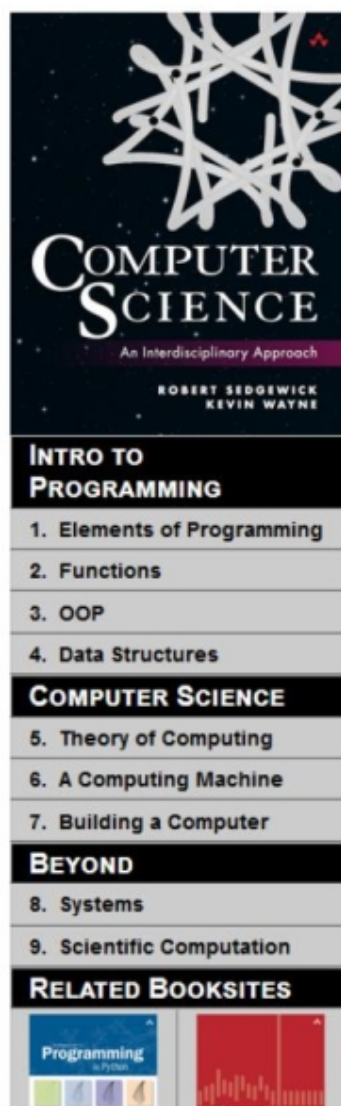
*a textbook for a first course in computer science  
for the next generation  
of scientists and engineers*

**Textbook.** Our textbook *Computer Science* [[Amazon](#) · [Pearson](#) · [InformIT](#)] is an interdisciplinary approach to the traditional CS1 curriculum with Java. We teach the classic elements of programming, using an "objects-in-the-middle" approach that emphasizes data abstraction. We motivate each concept by examining its impact on specific applications, taken from fields ranging from materials science to genomics to astrophysics to internet commerce.

The first half of the book (available separately as [Introduction to Programming in Java](#)) is organized around four stages of learning to program:

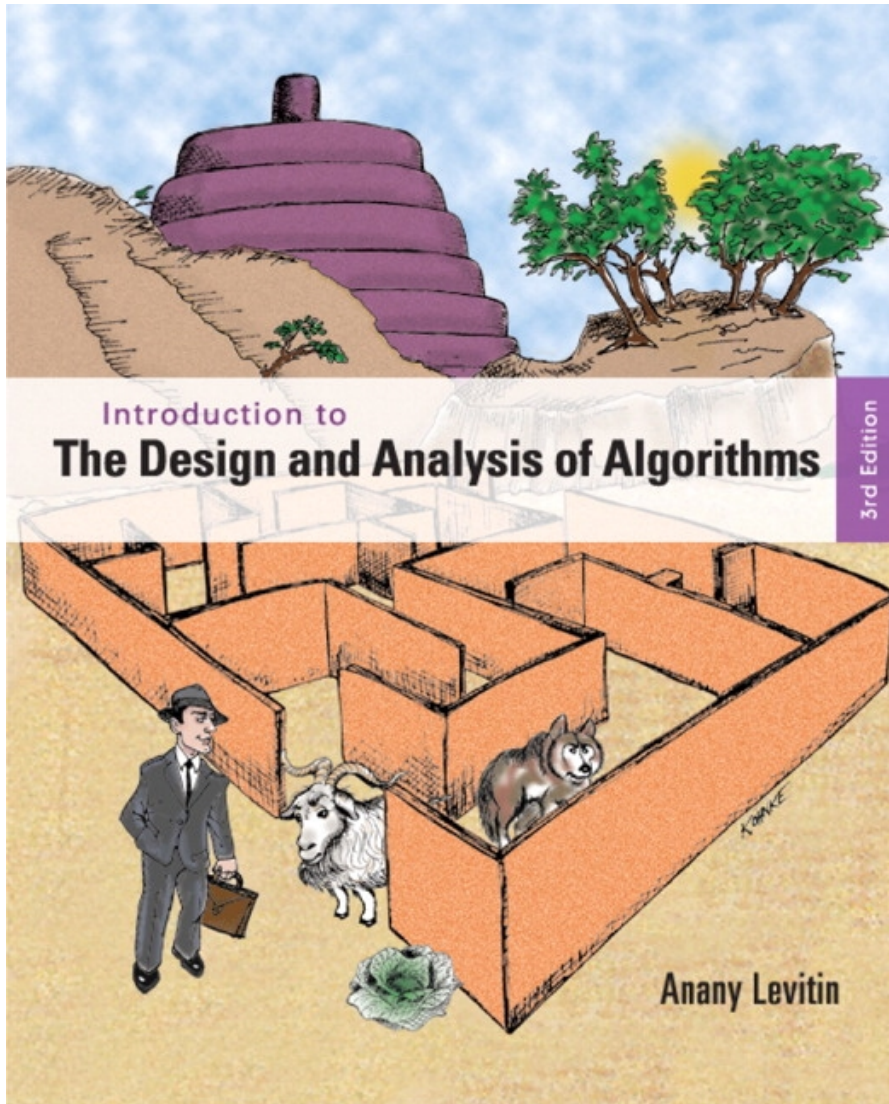
- [Chapter 1: Elements of Programming](#) introduces variables; assignment statements; built-in types of data; conditionals and loops; arrays; and input/output, including graphics and sound.
- [Chapter 2: Functions](#) highlights the idea of dividing a program into components that can be independently debugged, maintained, and reused.
- [Chapter 3: Object-Oriented Programming](#) emphasizes the concept of a data type and its implementation, using Java's class mechanism.
- [Chapter 4: Algorithms and Data Structures](#) discusses classical algorithms for sorting and searching, and fundamental data structures, including stacks, queues, and symbol tables.

The second half of the book explores core ideas of Turing, von Neumann, Shannon, and others that ignited the digital age.





# Reference B



# Course Evaluation:

- ▣ Class participation/quiz 10%
- ▣ Lab 20%
- ▣ Course project 20%
- ▣ Mid term exam 20%
- ▣ Final exam 30%

Enjoy Labs starting from the first week 😊

# Important

- ▶ Most ideas will be covered in class but some details might be skipped. These details are covered in the necessary reading.
- ▶ Assignments and project should be done individually although group discussions are allowed
- ▶ Any dishonest behaviour and cheating in assignments will be dealt with severely
- ▶ If you get an idea for a solution from others or online you must acknowledge the source in your submission



# Regulations on Academic Misconduct in courses for Undergraduate Students.docx

## Regulations on Academic Misconduct in courses for Undergraduate Students in the SUSTech Department of Computer Science and Engineering

From Spring 2022, the plagiarism policy applied by the Computer Science and Engineering department is the following:

- \* If an undergraduate assignment is found to be plagiarized, the first time the score of the assignment will be 0.
- \* The second time the score of the course will be 0.
- \* If a student does not sign the Assignment Declaration Form or cheats in the course, including regular assignments, midterms, final exams, etc., in addition to the grade penalty, the student will not be allowed to enroll in the two CS majors through 1+3, and cannot receive any recommendation for postgraduate admission exam exemption and all other academic awards.



- \* If an undergraduate assignment is found to be plagiarized, the first time the score of the assignment will be 0.**
- \* The second time the score of the course will be 0.**
- \* If a student does not sign the Assignment Declaration Form or cheats in the course, including regular assignments, midterms, final exams, etc., in addition to the grade penalty, the student will not be allowed to enroll in the two CS majors through 1+3, and cannot receive any recommendation for postgraduate admission exam exemption and all other academic awards.**

## What is OK, and what isn't OK?

It's OK to work on an assignment with a friend, and think together about the program structure, share ideas and even the global logic. At the time of actually writing the code, you should write it alone.

It's OK to use in an assignment a piece of code found on the web, as long as you indicate in a comment where it was found and don't claim it as your own work.

It's OK to help friends debug their programs (you'll probably learn a lot yourself by doing so).

**DO NOT Share Your Workable Code to Friends**

It's OK to show your code to friends to explain the logic, as long as the friends write their code on their own later.

It's NOT OK to take the code of a friend, make a few cosmetic changes (comments, some variable names) and pass it as your own work.



南方科技大学  
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

计算机科学与工程系  
Department of Computer Science and Engineering

## Undergraduate Students Declaration Form

This is \_\_\_\_\_ (student ID: \_\_\_\_\_, who has enrolled in \_\_\_\_\_ course of the Department of Computer Science and Engineering. I have read and understood the regulations on courses according to “Regulations on Academic Misconduct in courses for Undergraduate Students in the SUSTech Department of Computer Science and Engineering”. I promise that I will follow these regulations during the study of this course.

Signature:

Date:

# Lecture 1

- ▶ Course Introduction
- ▶ WarmUp: Packages
- ▶ WarmUp: Basic I/O APIs in Java from TextBook A
- ▶ Case Study: Union-Find, A Quick Tour on Algorithms & Data Structures

To be discussed in Week 2:

- ▶ Introduction to Algorithm Design and Analysis

# Packages

---

- Classes are grouped into packages
- Package names are dot-separated identifier sequences

```
java.util  
javax.swing  
com.sun.misc  
edu.sjsu.cs.cs151.alice
```

- Unique package names: start with reverse domain name

# Packages

---

- Add `package` statement to top of file

```
package edu.sjsu.cs.cs151.alice;  
public class Greeter { . . . }
```

- Class without package name is in "default package"
- Full name of class = package name + class name

```
java.util.ArrayList  
javax.swing.JOptionPane
```

# Importing Packages

---

- Tedious to use full class names
- `import` allows you to use short class name

```
import java.util.Scanner;  
.  
.  
.  
Scanner a; // i.e. java.util.Scanner
```

- Can import all classes from a package

```
import java.util.*;
```

# Importing Packages

---

- Cannot import from multiple packages

```
import java.*.*; // NO
```

- If a class occurs in two imported packages, `import` is no help.

```
import java.util.*;  
import java.sql.*;  
.  
.  
.  
java.util.Date d; // Date also occurs in java.sql
```

- Never need to import `java.lang`.



# Packages and Directories

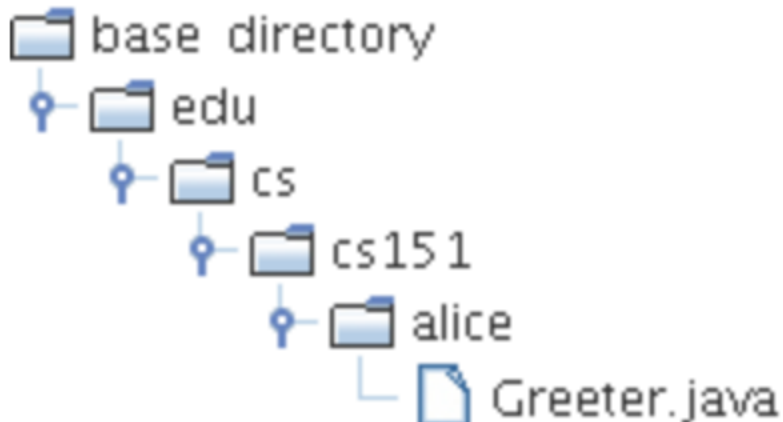
---

- Package name must match subdirectory name.

`edu.sjsu.cs.sjsu.cs151.alice.Greeter`

must be in subdirectory

*basedirectory/edu/sjsu/cs/sjsu/cs151/alice*



# Packages and Directories

---

- Always compile from the base directory

```
javac edu/sjsu/cs/sjsu/cs151/alice/Greeter.java
```

or

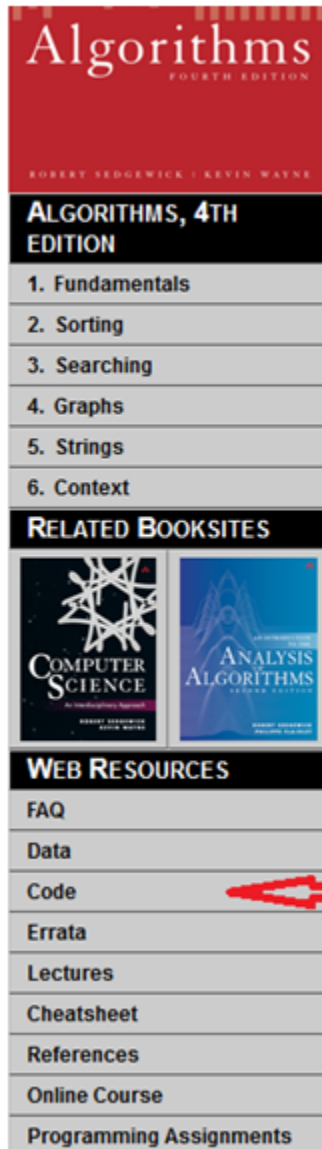
```
javac edu\sjsu\cs\sjsu\cs151\alice\Greeter.java
```

- Always run from the base directory


```
java edu.sjsu.cs.cs151.alice.GreeterTest
```

# WarmUp: Basic I/O APIs in Java from TextBook A

<https://algs4.cs.princeton.edu/code/>



**Online content.** This booksite contains tens of thousands of files, fully coordinated with our textbook and also:

- *Excerpts.* A condensed version of the text narrative, for reference while online.
- *Lectures.* Curated studio-produced online videos, suitable for remote instruction via [CUvids](#) .
- *Java code.* The algorithms and clients in this textbook, along with the standard libraries they use.
- *Exercises.* Selected exercises from the book and "web exercises" developed since its publication, along
- *Programming assignments.* Creative programming assignments that we have used at Princeton.

You can explore these resources via the sidebar at left.

**Textbook.** The textbook *Algorithms, 4th Edition* by Robert Sedgwick and Kevin Wayne [ [Amazon](#) · [Pearson](#) ] examining its impact on applications to science, engineering, and industry. The textbook is organized into six c

- *Chapter 1: Fundamentals* introduces a scientific and engineering basis for comparing algorithms and m
- *Chapter 2: Sorting* considers several classic sorting algorithms, including insertion sort, mergesort, and
- *Chapter 3: Searching* describes several classic symbol-table implementations, including binary search i
- *Chapter 4: Graphs* surveys the most important graph-processing problems, including depth-first search
- *Chapter 5: Strings* investigates specialized algorithms for string processing, including radix sorting, sub:
- *Chapter 6: Context* highlights connections to systems programming, scientific computing, commercial :

Reading a book and surfing the web are two different activities: This booksite is intended for your use while on when reinforcing your understanding of that material (for example, when reviewing for an exam).

**For teachers:**

**Standard input and output libraries.** We use these [standard input and output libraries](#) from *Introduction to Programming: An Interdisciplinary Approach*. They are part of [algs4.jar](#).

Reference A

§	PROGRAM	DESCRIPTION / JAVADOC
1.5 	StdIn.java 	read numbers and text from standard input
1.5 	StdOut.java 	write numbers and text to standard output
1.5 	StdDraw.java 	draw geometric shapes in a window
1.5 	StdAudio.java 	create, play, and manipulate sound
2.2 	StdRandom.java 	generate random numbers
2.2 	StdStats.java 	compute statistics
2.2 	StdArrayIO.java 	read and write 1D and 2D arrays
3.1 	In.java 	read numbers and text from files and URLs
3.1 	Out.java 	write numbers and text to files
3.1 	Draw.java 	draw geometric shapes
3.1 	DrawListener.java 	interactions with Draw
3.1 	Picture.java 	process color images
3.1 	GrayscalePicture.java 	process grayscale images
—	BinaryStdIn.java 	read bits from standard input
—	BinaryStdOut.java 	write bits to standard output
—	BinaryIn.java 	read bits from files and URLs
—	BinaryOut.java 	write bits to files

# Two ways to use these Basic I/O APIs:

- 1) Copy Java Code of APIs needed to your program folder, and Use them directly (**without package**).
- 2) Using the Textbook Library **algs4.jar** (next slide)

```
import edu.princeton.cs.algs4.MinPQ;  
import edu.princeton.cs.algs4.StdIn;
```

To be learned to use in details in Labs. 😊

- *Windows Command Prompt (temporary)*. Download [algs4.jar](#)  to, say C:\Users\username\algs4\algs4.jar. Compile and execute your program using the `-classpath` or `-cp` option.

```
javac -cp .;C:\Users\username\algs4\algs4.jar HelloWorld.java
java -cp .;C:\Users\username\algs4\algs4.jar HelloWorld
```

- *Windows Command Prompt (permanent)*. Download [algs4.jar](#)  to, say C:\Users\username\algs4\algs4.jar. Next, add `algs4.jar` to the `CLASSPATH` environment variable.

- *Windows 8 or 10: Control Panel → System and Security → System → Advanced system settings → Advanced → Environment Variables → CLASSPATH.*

*Windows 7: Start → Computer → System Properties → Advanced system settings → Environment Variables → User variables → CLASSPATH.*

*Vista: Start → My Computer → Properties → Advanced → Environment Variables → User variables → CLASSPATH.*

- Prepend the following to the *beginning* of the `CLASSPATH` variable:

```
C:\Users\username\algs4\algs4.jar;
```

The semicolons separate entries in the `CLASSPATH`.

- Click OK three times.

If you don't see a variable named `CLASSPATH`, click *New* and in the popup window enter `CLASSPATH` for the variable name. Then, perform the instructions above.

# Summary

- ▶ Course Introduction
- ▶ WarmUp: Packages
- ▶ WarmUp: Basic I/O APIs in Java from TextBook A
- ▶ Case Study: Union-Find, A Quick Tour on Algorithms & Data Structures (to be continued)

To be discussed in Week 2:

- ▶ Introduction to Algorithm Design and Analysis