



CS231: Fundamental Algorithms

Spring 2008

CLRS Reading: Chapter 1, Sections 2.1, 2.2, pages 1--27
Problem Set: Assignment #1 due Friday, February 8

A - 1



Administrativa

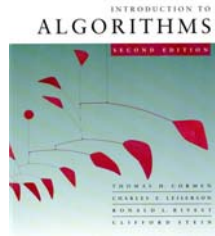
Instructor: Brian Tjaden
Meeting: Tuesdays and Fridays, 8:30 - 9:40am
Office Hours: Mondays 10:00am-11:30am,
Wednesdays 10:00-11:30am,
Thursdays 3:00pm-5:00pm
Drop-In Hours: Habiya Beg
FirstClass
Conference: CS231-S08

A - 2



CLRS




Introduction to Algorithms (2nd Edition)
by Thomas Cormen, Charles Leiserson,
Ronald Rivest, and Clifford Stein



A - 3



Course Materials

<p>Wellesley College CS 231 Spring 2008</p> <p>Course Home Page Policies Lecture Schedule & Slides Homework Assignments</p>	<div data-bbox="643 1310 735 1409" data-label="Image"></div> <h2>CS231: Fundamental Algorithms</h2> <p>Spring 2008</p> <hr/> <p><i>Instructor:</i> Brian Tinden <i>Meeting:</i> Tuesdays and Fridays, 8:30-9:40am in E111 <i>Office Hours:</i> TBD <i>FirstClass Conference:</i> CS231-508</p> <p>Welcome to CS231 <i>Computer Science 231</i> is an introduction to the design and analysis of fundamental algorithms in computer science. General techniques covered in the course include divide-and-conquer algorithms, dynamic programming, greediness, and probabilistic algorithms. Topics for the course include sorting, searching, graph algorithms, file compression, and NP-completeness. <i>Prerequisites:</i> CS230 and MATH225 <i>Distribution:</i> Mathematical Modeling <i>Semester:</i> Spring, Unit: 1.0</p> <p>Course Materials  CS231 course materials will be available on the course website as PDF files, which can be viewed and printed using the freely available Adobe Acrobat Reader program.</p> <p>Textbook  The text for the course is the second edition of <i>Introduction to Algorithms</i>, written by Thomas Cormen, Charles Leiserson, Ronald Rivest, and Clifford Stein, and published by MIT Press, Cambridge, Massachusetts. Several copies of the text, here after known as CLRS, are on reserve in the library.</p> <hr/> <p>Wellesley College Computer Science Department </p>
---	--

A - 4



Course Requirements

Problem Sets*	30%
Midterm Exam 1	20%
Midterm Exam 2	20%
Final Exam	30%
<hr/>	
Total	100%

*You are encouraged to work together. However, please acknowledge collaborative work.

A - 5



The Sorting Problem

Input: A sequence of n numbers $\langle a_1, a_2, \dots, a_n \rangle$.

Output: A permutation (reordering) $\langle a'_1, a'_2, \dots, a'_n \rangle$ of the input sequence such that $a'_1 \leq a'_2 \leq \dots \leq a'_n$.

A - 6

Sorting a Hand of Cards



A-7

Insertion Sort

U	N	S	O	R	T	E	D
U	N	S	O	R	T	E	D
N	U	S	O	R	T	E	D
N	S	U	O	R	T	E	D
N	O	S	U	R	T	E	D
N	O	R	S	U	T	E	D
N	O	R	S	T	U	E	D
E	N	O	R	S	T	U	D
D	E	N	O	R	S	T	U

A-8

Insertion Sort Pseudocode*

```

InsertionSort(A)
1   for j ← 2 to length[A]
2       do key ← A[j]
3           » Insert A[j] into the sorted
              sequence A[1..j-1]
4           i ← j-1
5           while i > 0 and A[i] > key
6               do A[i+1] ← A[i]
7                   i ← i-1
8           A[i+1] ← key
    
```

*Be sure to read the authors' list of pseudocode conventions in section 2.1 of the text.

A - 9

Correctness of Insertion Sort

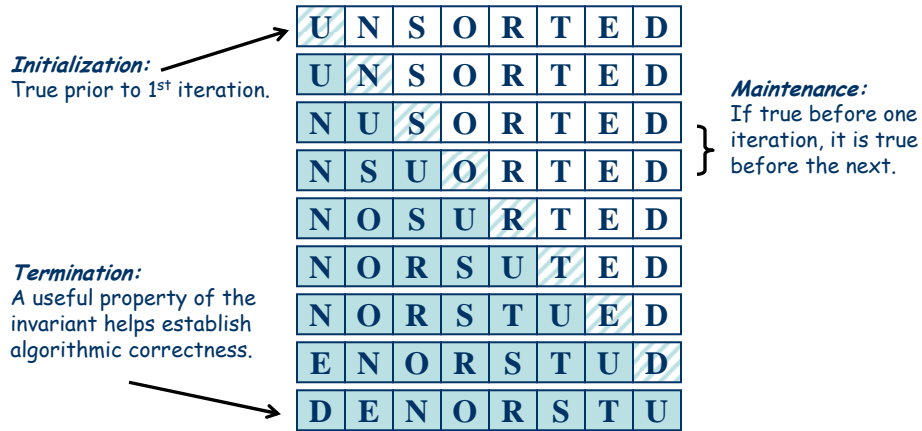
U	N	S	O	R	T	E	D
U	N	S	O	R	T	E	D
N	U	S	O	R	T	E	D
N	S	U	O	R	T	E	D
N	O	S	U	R	T	E	D
N	O	R	S	U	T	E	D
N	O	R	S	T	U	E	D
E	N	O	R	S	T	U	D
D	E	N	O	R	S	T	U

Loop Invariant:

At the start of each for loop, subarray $A[1..j-1]$ consists of the elements originally in $A[1..j-1]$ but in sorted order.

A - 10

Using Insertion Sort's Loop Invariant*



*At the start of each **for** loop, subarray $A[1..j-1]$ consists of the elements originally in $A[1..j-1]$ but in sorted order.

A - 11

Analysis of Algorithms

- How do we evaluate (and compare) different algorithmic solutions to the same problem?
- On what input do we evaluate an algorithm's performance?

We make certain assumptions...
For instance, we use a *random-access machine* (RAM) model of computation.

A - 12



Analysis of Insertion Sort

		<i>cost</i>	<i>times</i>
	InsertionSort(A)		
1	for $j \leftarrow 2$ to $length[A]$	1	
2	do $key \leftarrow A[j]$	1	
3	» Insert $A[j]$ into the sorted sequence $A[1..j-1]$	0	
4	$i \leftarrow j-1$	1	
5	while $i > 0$ and $A[i] > key$	1	
6	do $A[i+1] \leftarrow A[i]$	1	
7	$i \leftarrow i-1$	1	
8	$A[i+1] \leftarrow key$	1	

A-13