CSE 310
Data Structures and Algorithms

Instructor: Hanghang Tong
Assistant Professor
CIDSE

Course materials available through Blackboard on MyASU

Arizona State University

Lecture Overview

- Discussion on syllabus and logistics
  - Prerequisites, topics to cover, assessment
- Introduction to Algorithms & Data Structures
  - What is an algorithm; What do we care about an algorithm
  - Examples of algorithms

About This Course

- This is a core course of our CSE programs.
  - It is typically viewed as one of the most difficult courses by CSE students here.
    - Your being a straight-A student *thus far* does not mean you can have an easy pass, let alone an easy A in this course.
  - But it is also one of those few courses that makes you distinctively "computer science"-ish (versus other engineering majors)
  - Try to do this yourself afterwards: search on the Internet about "Google interview questions"

Staff

- Instructor: Hanghang Tong
  - Doing research on data mining, big data analytics → all involving designing sophisticated algorithms
  - Office: BY416, Office Hours: M/W 10-11am or other times by appointment.
- TAs: (All the office hours by TAs will be at CenterPoint 114)
  - Joydeep Banerjee (joydeep.banerjee@asu.edu), OH: Tues/Thurs 1-2pm
  - Jun Shen (jun.shen.1@asu.edu), OH: Tues 10-11am, Wedn 1-2pm
  - Huijun Wu (huijun.wu@asu.edu), OH: Mon 1:15-2:15pm, Wedn: 10:30-11:30am
- Grader: Rongyu Lin (No OH)

➢ Between the TAs and me, we have a total of 8 hours of office hours per week for helping you outside the class hours.

➢ Additional Resource for Help: Discussion Board on BB.

Prerequisites

- For CS/CSE students, grades of at least C in CSE 220 or CSE 240 and at least D in MAT 243. For CMS students, grades of at least C in CSE 210 and at least D in MAT 243 or MAT 300. Programming experience in C or C++ is expected.
  - Read the Appendix A of the textbook: the appendix should read like a review of known materials rather than new knowledge to you (except perhaps C, which is on probabilities, but you should at least be taking a probability course now).
  - Try the self-review test in BB
    - *Otherwise, this course may not be right for you.*

- Proficiency in C/C++ programming is required
  - Class project will require programming in C/C++, in Linux.
  - This course will NOT teach you how to program in C/C++; You are assumed to know that already. *Otherwise, this course may not be right not for you.*

Course Information

- Course materials will be available only through the Blackboard on MyASU.

- Required textbooks:

  – ***Introduction to Algorithms***, Cormen, T. H., Leiserson, C.E., Rivest, R.L., and Stein, C., The MIT Press, Cambridge Massachusetts, 3rd edition, 2009.

  ➢ This is a ***required*** book. You need to have it handy immediately. Cannot use "no book" as an excuse for late submission of assignments.
    - I'm not aware of the existence of a "desk copy" either. ➔ ***Purchase your copy now if you haven't done so.***

  ➢ It is a book worth keeping as a reference for your job interviews and even your future jobs.

Course Information

- Lecture notes will be posted before each class
  - May be updated throughout the semester. You need to update your version accordingly.
  - **Lecture notes may NOT include examples that are worked out on the whiteboard.**
  - **Good attendance is key for efficient and effective study**

- Lecture Note Takers:

A notetaker is needed for this class. In exchange for providing this service, the selected notetaker will be paid a stipend of $25.00 per credit, i.e. total of $75 for a 3 credit class OR if preferred, awarded a letter of community service at the end of the semester. If you take clear, concise notes and are willing to share a copy, please follow up with me after class to receive instructions on how to sign up online.

Course Information

• Homework assignments (including the class projects) and supplemental reading materials will all be posted on the Blackboard.

  – **Late submissions of the assignments will not be accepted.**

•Again, the class projects will require programming work. All programming work needs to be completed in C/C++, in Linux

  –Details will be announced soon on BB

Assessment

- Homework 10%: 4 of equal weight.
    - ➢No late submission will be accepted.
- Projects: 15%, 3 equal weight, all in c/c++ only on general.asu.edu (Linux)
    - ➢No late submission will be accepted.
- Midterm Exams 30% : there will be two of them, closed book.
    - –Each exam covers only the period from the previous exam until the current exam.
    - – The two exams amount to 30% total (equal weight).
- Final Exam 45%: 9:50-11:40am, December 7th, comprehensive, closed book
- **No make-up exam will be given unless it is for genuine, verifiable emergency.**

Assessment – Key Dates

Assignments	Out	Due	Projects	Out	Milestone	Due	Midterms	Scheduled
A1	08/24	09/11	P1	08/24	09/09	09/23	M1	09/30
A2	09/16	10/09	P2	09/23	10/16	10/28	M2	11/09
A3	10/14	11/06	P3	10/28	11/20	12/04		
A4	11/06	11/25						

Assessment

- The following cutoffs represent what will be *likely* used to generate the letter grades:

A+ >= 97%	**A >=90% & < 97%**	**A- >=87% & < 90%**
B+ >= 84% & < 87%	**B >=80% & < 84%**	**B- >=77% & < 80%**
C+ >= 74% & < 77%	**C >=70% & < 74%**	**C- >=67% & <70%**
	D >= 60%& <67%	**E < 60%**

- The above cutoffs are tentatively and may be adjusted *slightly*; However, there will be *no general curve-fitting* in assigning the final grades.

Major Topics & Tentative Schedule

1	**Syllabus & Intro, Asymptotic Notation,**
2	**Merge Sort, Quick Sort**
3	**Heaps, Priority Queues**
4	**Sorting Lower Bound, Linear sort algorithms (briefly); Selection**
5	**Selection; Hash Tables**
6	**Hash Tables; Binary search tree**
7	**Red-Black Trees**
8	**Disjoint Sets**
9	**Matrix Multiplication**
10	**Longest common sequence; Depth First Search**
11	**Breadth First Search; Topological Sort & SCC**
12	**Minimum Spanning Trees; Shortest Paths**

Roughly, each line corresponds to 1 week.

Questions or Comments

Introduction to Algorithms

- What is an algorithm: An algorithm is a well-defined computational procedure that takes some input, produces some output, and then terminate.

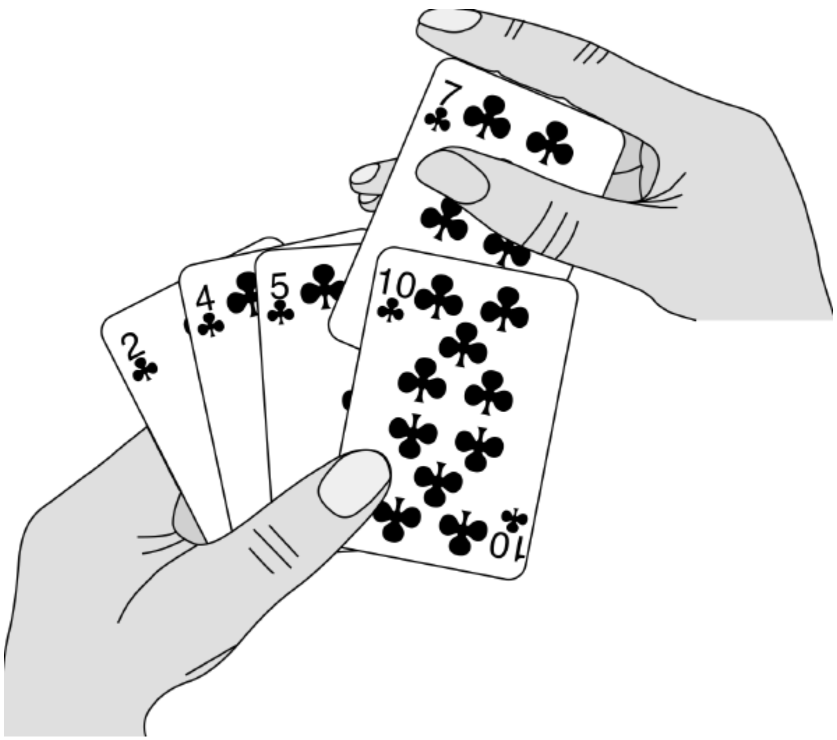
- An example of an algorithm: design a computational procedure for sorting a given set of *n* number. E.g., n = 5 for the example below.

  Given input: 5, 2, 4, 10, 7

  Desired output: 2, 4, 5, 7, 10

  Your fancy (computational) way of generating the desired output for any given input ➔ An algorithm

Analogy to sorting cards …



- Think about what you would do

- There may be many ways, but one intuitive *procedure* is to always *insert* a card into the right place

  – Works naturally when you draw a stack of cards one by one and put them into your hand.

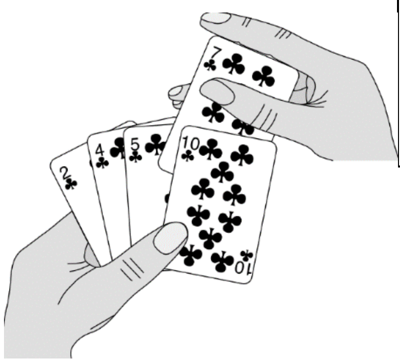
  – Similarly, if you sort all cards in your hand → "*in place*" sorting

Computationally represent the procedure

- Pseudocode: Liberal use of English; Use of indentation for block structure; Omission of error handling (e.g., check if an array is empty).

```
//  Pseudocode for Insertion Sort

INSERTION-SORT(A)
for j = 2  to  A.length     // A.length = n
        key = A[j]
        // Insert A[j] into sorted sequence A[1 … j-1]
        i = j – 1
        while i > 0  and  A[i] > key
                A[i+1] = A[i]
                i = i – 1
        A[i+1] = key
```



So, how good is it?

- Does it work? Is it fast enough? Does it require a lot of memory?

  ➔ Analysis of algorithms.

Correctness

Efficiency → Time complexity

→ Space complexity

We want to answer these questions independent of machines, programming languages, etc.

➔ RAM (Random Access Machine): Model of computation

- Basic instruction can be performed in one unit of time (time step).

Complexity of An Algorithm

- Time complexity: Running time of an algorithm in terms of the size of the input to the algorithm.
- Space complexity: measured using the number of memory units required for executing the procedure.
- Both time complexity and space complexity are in general a function of the input size, typically denoted n.
  - Sorting 100 elements takes more time, as well as requires more memory, than sorting 3 elements

Let's analyze our Insertion Sort algorithm

-- First look at how it works on a specific input.

Example:

3	9	6	1	2	9 should be inserted after 3 – no change
3	9	6	1	2	6 should be inserted between 3 and 9
3	6	9	1	2	1 should be inserted before 3
1	3	6	9	2	2 should be inserted between 1 and 3
1	2	3	6	9	

times	cost
n	c_1
n-1	c_2
0	c_3=0
n-1	c_4
$\Sigma^n_{j=2}\, t_j$	c_5
$\Sigma^n_{j=2}(t_j\text{-}1)$	c_6
$\Sigma^n_{j=2}(t_j\text{-}1)$	c_7
n-1	c_8

```
//  Pseudocode for Insertion Sort

INSERTION-SORT(A)
for j = 2  to  A.length     // A.length = n
        key = A[j]
        // Insert A[j] into sorted sequence A[1 ... j-1]
        i = j – 1
        while i > 0  and  A[i] > key
                A[i+1] = A[i]
                i = i – 1
        A[i+1] = key
```

t_j = the number of times the while loop test is executed for that value of j.

t_j *-1* = # of elements in A[*1* ...*j*-*1*] that are > A[*j*]

For the insertion sort in the previous pages, its running time is:

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum^{n}_{j=2} t_j + c_6 \sum^{n}_{j=2} (t_j - 1)$$
$$+ c_7 \sum^{n}_{j=2} (t_j - 1) + c_8(n-1)$$

$T(n)$ depends not just on n, but also the kinds of input arrays:

BEST-CASE: array is sorted.

In this case, $t_j = 1$ for all j

The running time becomes:

$$c_1 n + c_2(n-1) + c_4(n-1) + c_5(n-1) + c_8(n-1)$$
$$= (c_1 + c_2 + c_4 + c_5 + c_8) n - (c_2 + c_4 + c_5 + c_8)$$

$\rightarrow$ A linear function of n.

WORST-CASE: array is in reverse order

In this case, $t_j = j$ for all j

The running time becomes:

$c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum^{n}_{j=2} j + c_6 \sum^{n}_{j=2}(j-1) + c_7 \sum^{n}_{j=2}(j-1) + c_8(n-1)$

$= c_1 n + c_2(n-1) + c_4(n-1) + c_5[n(n+1)/2 - 1] + c_6[\, n(n-1)/2 \,]$

$\quad + c_7 [\, n(n-1)/2 \,] + c_8(n-1)$

$= (c_5/2 + c_6/2 + c_7/2)\, n^2 \quad + (c_1 + c_2 + c_4 + c_5/2 - c_6/2 - c_7/2 + c_8)\, n - (c_2 + c_4 + c_5 + c_8)$

➔ A quadratic function of n.

Note: $\sum^{n}_{j=2} j = n(n+1)/2 - 1$ and $\sum^{n}_{j=2}(j-1) = n(n-1)/2$

--- See Appendix A of the textbook for more summation.

- What's the big deal about the difference between "a linear function" and "a quadratic function" of n?
  - $n = 10$ ➔ $n^2 = 100$, $n = 1000$ ➔ $n^2 = 1{,}000{,}0000$, …
- There are worse algorithms whose complexity may be of n^3 or even 2^n
  - The difference will be even more! ➔ Eventually render some algorithms with a high complexity practically useless except for very small n.

Data Structure

- The course title contains the term “Data Structure”: why is it there?

- A data structure is a way to store and organize data in order to facilitate access and modifications.
  - Think about how you would compile a dictionary (to facilitate looking up any entry)
  - Think about how you would organize all songs on your PC (to facilitate retrieving a song of your choice)

- Efficient algorithms are often designed together with accompanying data structures
  - Graph-based algorithms

Summary & Reading Assignment

- What is an algorithm?
- Some basic ideas in analyzing an algorithm
- An examples: insertion sort

  ➔Read to review: Chapter 1, Chapter 2.1, 2.2

- Next time: formal analysis techniques: asymptotic notations

  ➔ Read to prepare: Chapter 3