Design and Analysis of Algorithms

CSE 5311

Lecture 1 Administration & Introduction

Junzhou Huang, Ph.D.

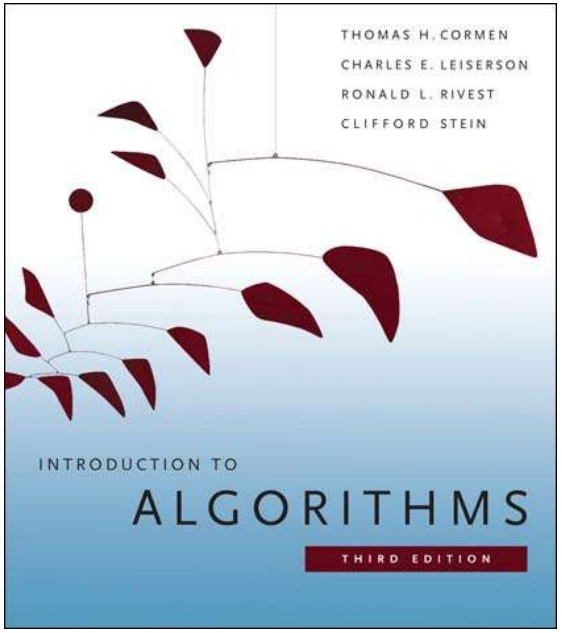
Department of Computer Science and Engineering

Administration

- **Course CSE5311**
  - What: Design and Analysis of Algorithms
  - When: Friday 1:00 ~ 3:50pm
  - Where: Online
  - Who: Junzhou Huang (Office ERB 650) jzhuang@uta.edu
  - Office Hour: Friday 3:50 ~ 5:50pm and/or appointments
  - Homepage: http://ranger.uta.edu/~huang/teaching/CSE5311.htm
    (**You're required to check this page regularly**)
- **Lecturer**
  - PhD in CS from Rutgers, the State University of New Jersey
  - Research areas: machine learning, computer vision, medical imaging informatics
- **GTA**
  - Chunyuan Li (Office ERB 426), chuanyuan.li@mavs.uta.edu
  - Office hours: Wed. 9:00am ~ 12:00pm and/or appointments

Study Materials

- **Prerequisites**
  - Algorithms and Data Structure (CSE 2320)
  - Theoretical Computer Science (CSE 3315)
  - What this really means:
    - You have working experience s on software development.
    - You know compilation process and programming
    - Elementary knowledge of math and algorithms
- **Text book**
  - Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein, **Introduction to Algorithms**, third edition
  - https://mitpress.mit.edu/books/introduction-algorithms



Study Materials

- **Text book**
  - We will not cover all the chapters of the book
  - We will not cover all sections of the covered chapters
  - We will not fully follow the order of the book
  - The contents uncovered in slides/lectures are optional
- **Recommended Reference**
  - Robert Sedgewick, **Algorithms in C**. Addison - Wesley, 1990

- **Acknowledgments**
  - Class notes partially based on 5311 classes taught at UTA in prior years
  - Material from textbook site
  - Lots of material available on the web (via google search, wikipedia)

Grading

- **Distribution**
  - 20% Projects
  - 35% Midterm Exam
  - 40% Final Exam
  - 5% Class Participation

    100%
- **Attention**
  - Homework is as important as any other aspects of your grade!
  - Attendance though not mandatory, but is HIGHLY encouraged.
  - The university makeup policy will be strictly adhered to. Generally, no make-up exams/quizzes except for university sanctioned reasons.
  - When missing an exam/quiz due to unavoidable circumstances, PLEASE notify the instructor and request a makeup approval **ahead of time**.

Final Grade

- **Final Letter Grade**
  - [90 100] --- A
  - [80 90) --- B
  - [70 80) --- C
  - [60 70) --- D
  - [00 60) --- F
- **Attention**
  - Final letter grades will be assigned based on absolute percentage
  - [ ] denotes inclusion and ( ) denotes exclusion.
  - The instructor reserves the right to move the thresholds down based on the distribution of final percentages, but they will not move up.

Assignments

- **Homework assignments**
  - Assigned in class, typically due one week later at the start of next lecture
  - Automatic 20% deduction for each day late
  - Homework is not accepted more than 3 days late
- **Projects**
  - Assigned in class, typically due 1~3 weeks later after assignments
  - They are not created equally.
- **Collaboration**
  - You may discuss assignments with others, but must write up them individually. Please identify collaborators on your assignment cover sheet
  - Failure to comply with this policy is a violation of academic integrity
- **Start early! Start early! Start early !!!**

Information

- **Course Webpage**
  - Check the web page regularly (2 times per week).
  - Announcements, assignments, and lecture notes will be posted there.
- **Grade Appeal**
  - You may appeal the grade in writing (email) within 5 class days.
  - Appealed to the appropriate GTA firstly, then to the instructor, if necessary.
  - Please refer to the UTA Catalog for the detailed guide of grade appeals.
- **Drop Policy**
  - The university withdrawal policy will be strictly adhered to.
- **Others**
  - Accommodating students with disabilities
  - Student Support Services
  - Etc.

Questions

Course Overview

- **What is it?**
  - Algorithms
  - Design and Analysis
- **Why is a CS course?!?**
  - The key of the computer science
- **Will I really ever use this stuff again?**
  - Definitely, analysis and design
  - Necessary knowledge for a CS student
  - You may not become a professional algorithm designer but you definitely need know how to analysis and design the algorithms for the problems in your future career and even in your life
- **How to succeed in this course?**

Why Are You In This Class?

- **Something interesting about you**
  - Why you picked your major?
  - Life Plan
- **To learn background in order to take more advanced classes in computer science and engineering**
  - Database, big data analytics, compiler, Computer Network, Embedded Systems, artificial intelligent, machine learning, data mining, computer vision, etc.
- **Understand the effect of an algorithm on the code you write or read**
  - Learn how to efficiently use and control the computer
- **To have the necessary background to understand innovations in intelligent design or related others**
  - Your desktop, laptop, ipad, iphone, google search, facebook social network, etc.
- **Necessary to become a professional algorithm designer?**
  - But to be able to read and understand
  - To be able to understand innovative ideas

What?

- ***The theoretical study of design and analysis of computer algorithms***
- **Basic goals for an algorithm**
  - Always correct
  - Always terminates
- **Our class: performance**
  - Performance often draws the line between what is possible and what is impossible.
- **Design and Analysis of Algorithms**
  - **Analysis:** predict the cost of an algorithm in terms of resources and performance
  - **Design:** design algorithms which minimize the cost

Machine Model

- **Generic Random Access Machine (RAM)**
  - Executes operations sequentially
  - Set of primitive operations: Arithmetic. Logical, Comparisons, Function calls
- **Simplifying assumption**
  - All operations cost 1 unit
  - Eliminates dependence on the speed of our computer
  - Otherwise impossible to verify and to compare

The Problem of Sorting

Input: sequence $\langle a_1, a_2, \ldots, a_n \rangle$ of numbers.

Output: permutation $\langle a'_1, a'_2, \ldots, a'_n \rangle$ such that $a'_1 \le a'_2 \le \cdots \le a'_n$.

Example:

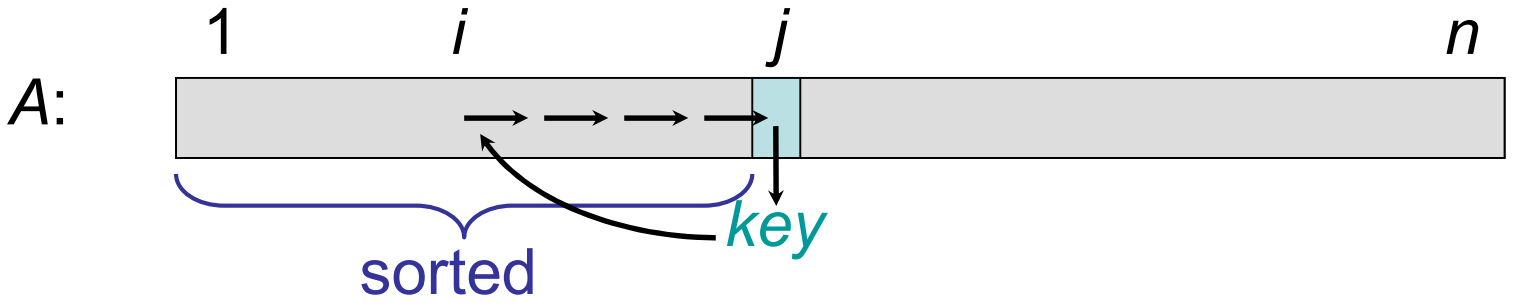
Input: 8 2 4 9 3 6

Output: 2 3 4 6 8 9

Insertion sort

"pseudocode"

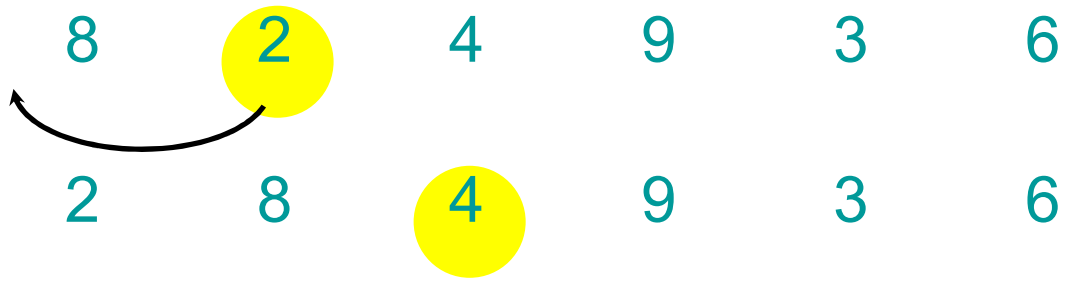
```
INSERTION-SORT (A, n)        ▷ A[1 . . n]
    for j ← 2 to n
        do key ← A[ j]
           i ← j – 1
           while i > 0 and A[i] > key
               do A[i+1] ← A[i]
                  i ← i – 1
           A[i+1] = key
```



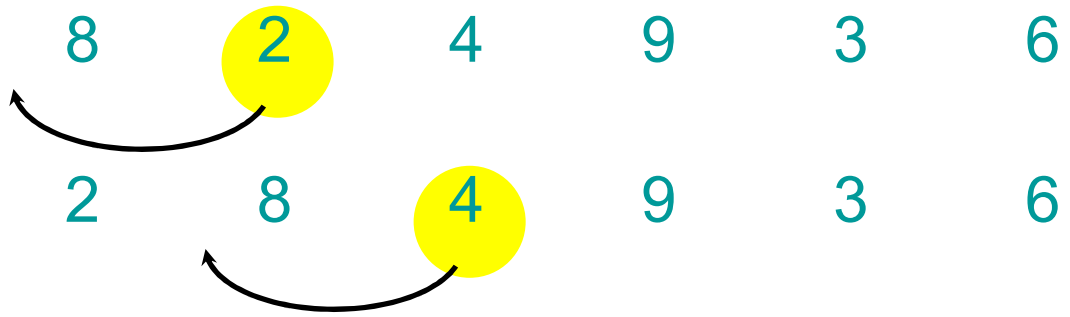
Example of Insertion Sort

Example of Insertion Sort

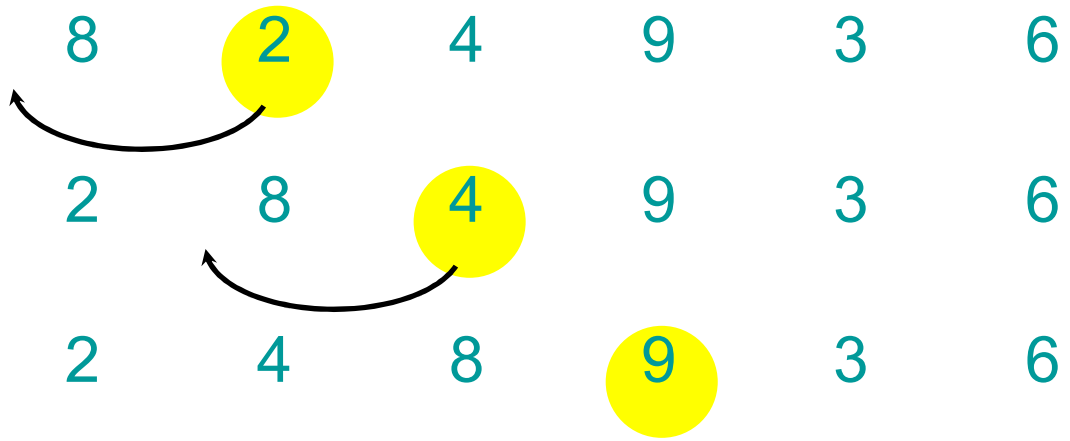
Example of Insertion Sort



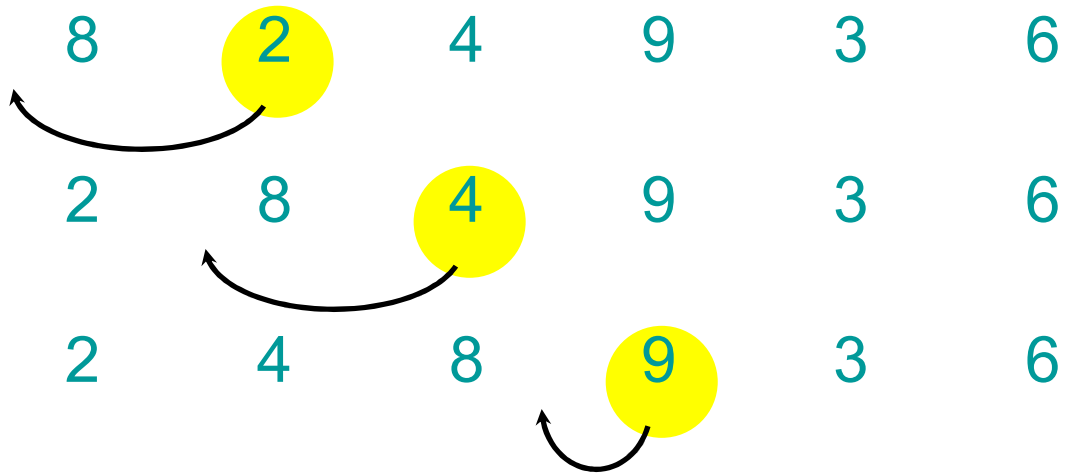
Example of Insertion Sort



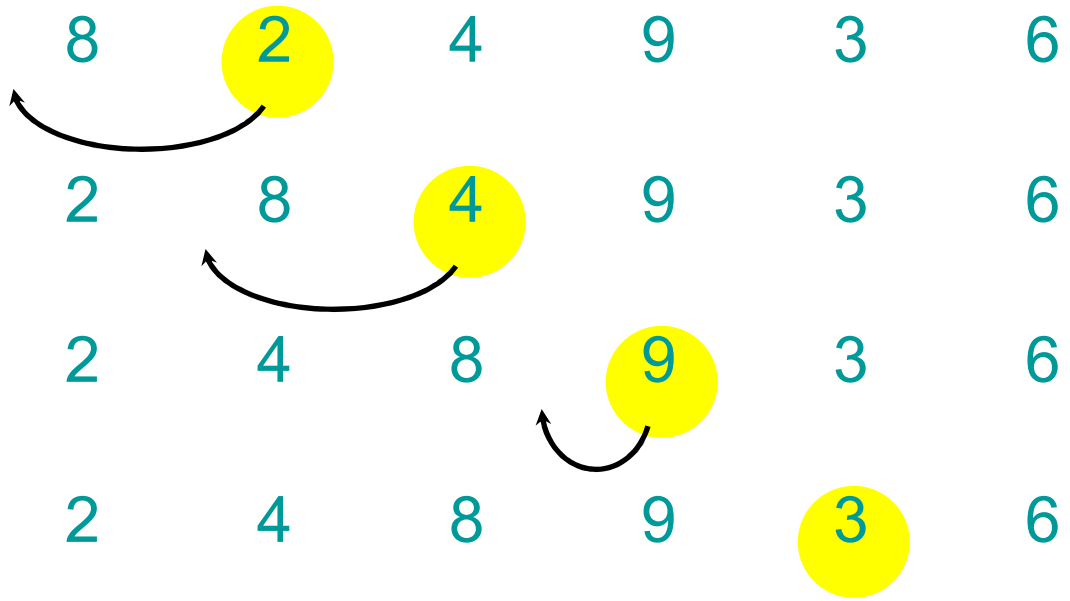
Example of Insertion Sort



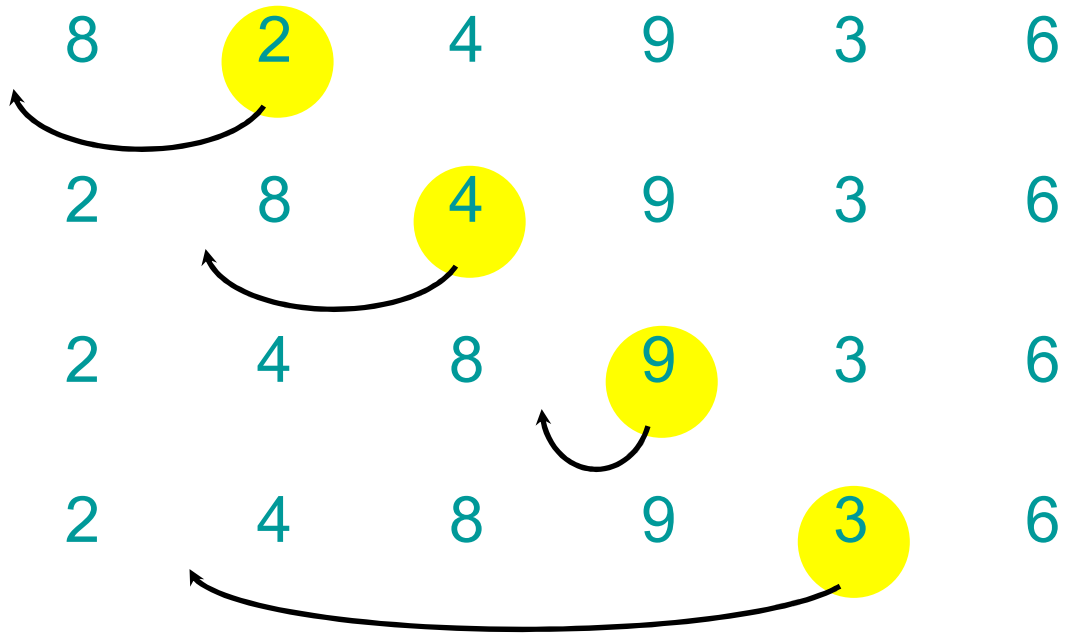
Example of Insertion Sort



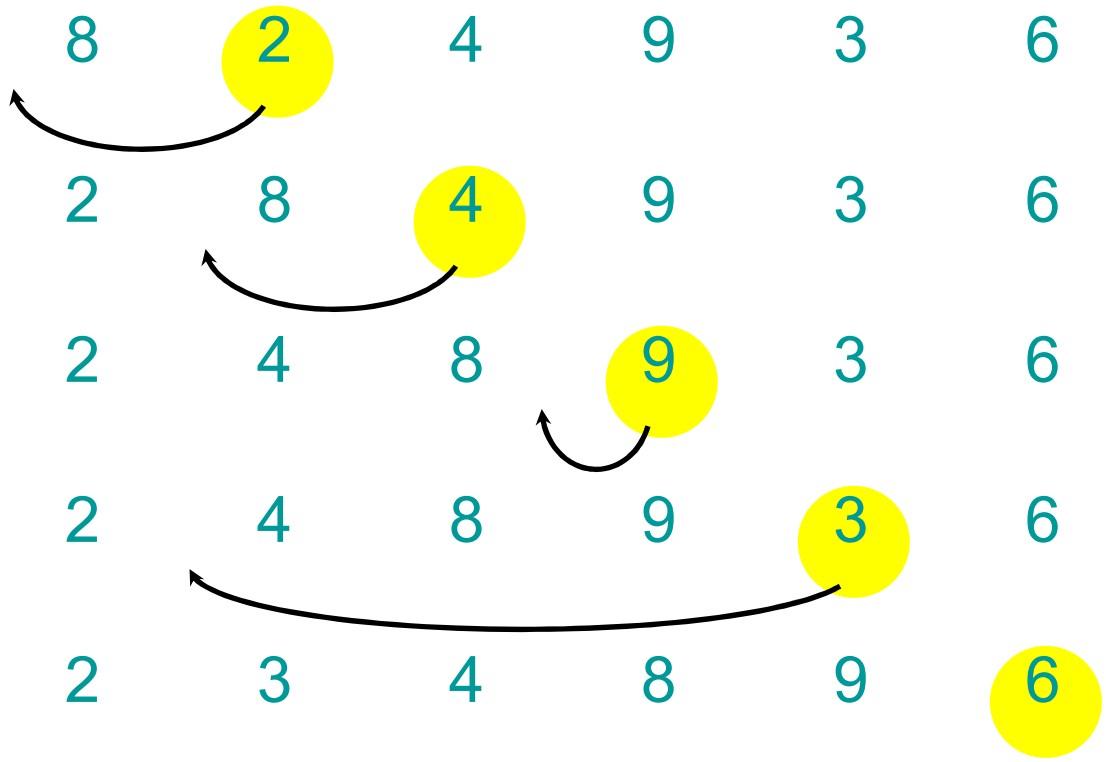
Example of Insertion Sort



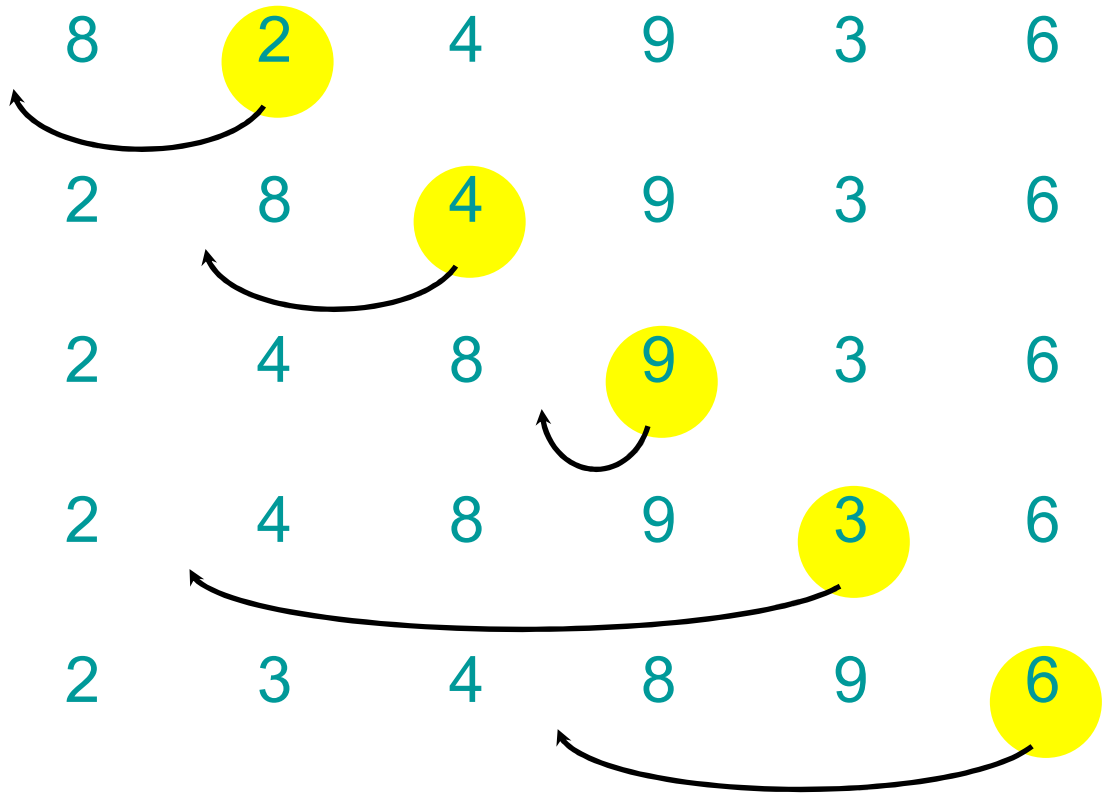
Example of Insertion Sort



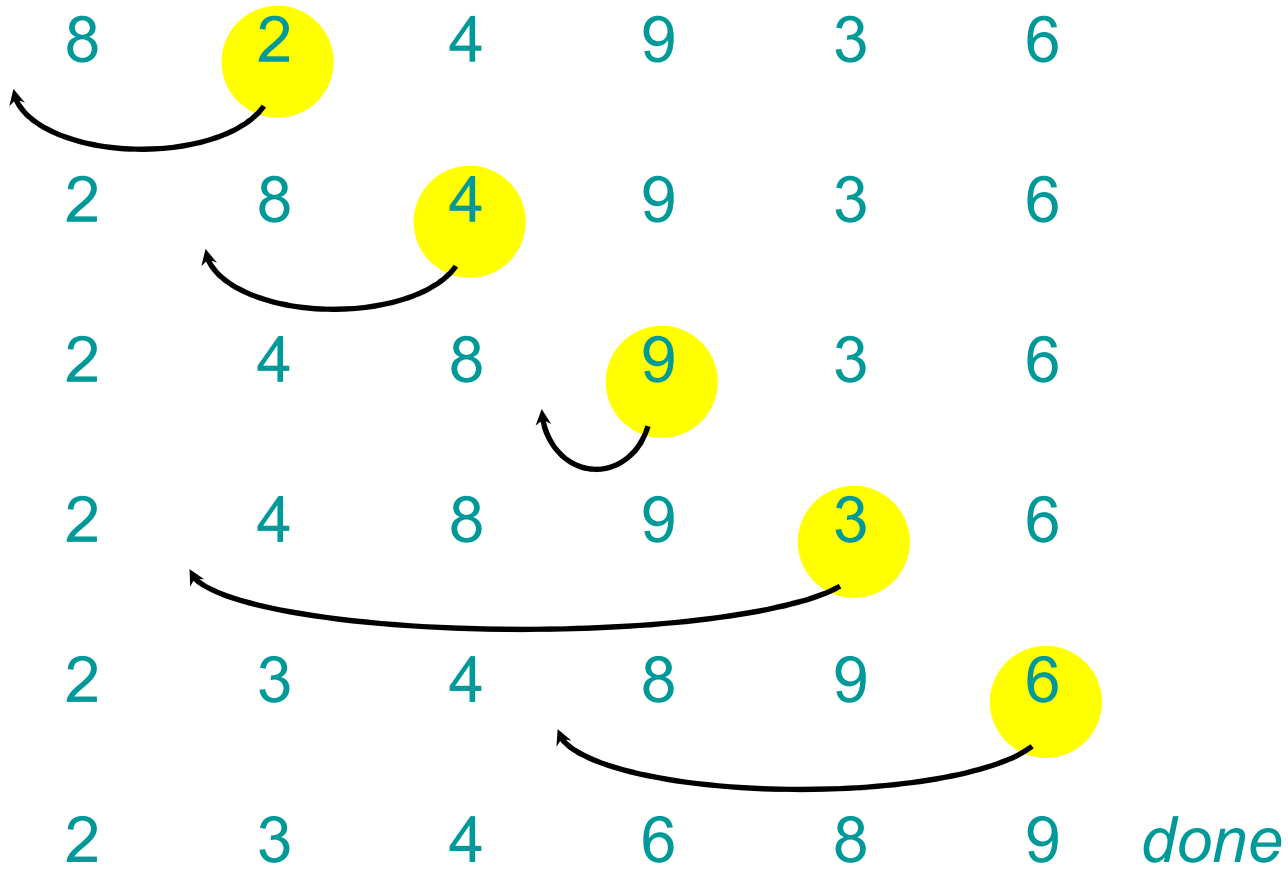
Example of Insertion Sort



Example of Insertion Sort



Example of Insertion Sort



Running Time

- **Running Time**
  - Depends on the input
  - An already sorted sequence is easier to sort.
- **Major Simplifying Convention**
  - Parameterize the running time by the size of the input, since short sequences are easier to sort than long ones.
  - $T_A(n)$ = time of A on length n inputs. Generally, we seek upper bounds on the running time, to have a guarantee of performance.
- **Kinds of analyses**
  - **Worst-case:** (usually) T(n) = maximum time of algorithm on any input of size n
  - **Average-case:** (sometimes) T(n) = expected time of algorithm over all inputs of size n. Need assumption of statistical distribution of inputs.
  - **Best-case:** (Never) Cheat with a slow algorithm that works fast on some input.

Measuring Algorithm Complexity

- **How long does it take to execute a program?**
  - Efficient algorithm, the better
- **How long does it take to go from point A to Point B**

- **Need to know:**
  - Speed: Walk/driving
  - Distance: variable
  - Traffic condition
  - Walk?
  - Driving?

Analysis

- **Simplifications**
  - Ignore actual and abstract statement costs
  - Order of growth is the interesting measure:
    Highest-order term is what count
    Doing asymptotic analysis
    As the input size grows larger it is the high order term that dominates

- **Teaching Goals**
  - Show that by knowing more about the underlying algorithm design and analysis, one can be more effective as a computer scientist or engineer.

Upper Bound Notation

- **Definition**
  - In general, a function f(n) is O(g(n)) if there exist positive constants c and n_0 such that $f(n) \le c \cdot g(n)$ for all $n \ge n_0$
  - Formally, $O(g(n)) = \{ f(n): \exists$ positive constants c and n_0 such that $f(n) \le c \cdot g(n)$ $\forall\ n \ge n_0$

- **A polynomial of degree k is $O(n^k)$**
- **Proof:**

  Suppose $f(n) = b_k n^k + b_{k-1} n^{k-1} + \ldots + b_1 n + b_0$

  Let $a_i = | b_i |$

  $f(n) \le a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0$

$$\le \quad n^k \sum a_i \frac{n^i}{n^k} \quad \le \quad n^k \sum a_i \quad \le \quad cn^k$$

Upper Bound Notation

- **We say InsertionSort's run time is $O(n^2)$**
  - Properly we should say run time is *in* $O(n^2)$; Read O as "Big-O"
- **Proof**
  - Suppose runtime is $an^2 + bn + c$;
  - If any of a, b, and c are less than 0 replace the constant with its absolute value
  - $an^2 + bn + c \leq (a + b + c)n^2 + (a + b + c)n + (a + b + c)$

    $\leq 3(a + b + c)n^2$

    for $n \geq 1$; Let $c' = 3(a + b + c)$ and let $n_0 = 1$
- **Questions**
  - Is InsertionSort $O(n^3)$?
  - Is InsertionSort $O(n)$?

Lower Bound Notation

- **Definition**
  - In general a function f(n) is $\Omega(g(n))$ if $\exists$ positive constants c and n_0 such that $0 \leq c \cdot g(n) \leq f(n) \quad \forall\ n \geq n_0$
- **We say InsertionSort's run time is $\Omega(n)$**
- **Proof**
  - Suppose run time is $a \cdot n + b$
  - Assume a and b are positive (what if b is negative?)

$a \cdot n \leq a \cdot n + b$

Asymptotic Tight Bound

- **Asymptotic Tight Bound Θ**
  - A function f(n) is Θ(g(n)) if ∃ positive constants c_1, c_2, and n_0 such that $c_1\, g(n) \le f(n) \le c_2\, g(n)$, $\forall\ n \ge n_0$
  - **Theorem:** f(n) is Θ(g(n)) iff f(n) is both O(g(n)) and Ω(g(n))
  - **Proof:** self practice
- **Other Asymptotic Tight Bounds**
  - A function f(n) is o(g(n)) if ∃ positive constants c and n_0 such that $f(n) < c\, g(n)$ $\forall\ n \ge n_0$
  - A function f(n) is ω(g(n)) if ∃ positive constants c and n_0 such that $c\, g(n) < f(n)$ $\forall\ n \ge n_0$
  - Intuitively,
    - o() is like <
    - ω() is like >
    - Θ() is like =
    - O() is like ≤
    - Ω() is like ≥

Course Goals

- **Teaching Style**
  - Algorithm courses have been traditionally taught by following textbooks and covering basic concepts and algorithms for different problems.
  - We're going to follow this style and additionally cover some fresh problems and algorithms appeared in FLG interviewing.
- **Teaching Goals**
  - Show that by knowing more about the underlying algorithm design and analysis, one can be more effective as a computer scientist or engineer.
  - Write programs that are more reliable and efficient for the specific goal.
  - Understand how program performance depends on underlying factors including the designed algorithm.
  - Learn how to implement an effective and efficient software system according to the request and the available resources (memory, communication, disk, GPU, etc.)

Course Expectation

- **What to expect from the course:**
  - Will cover key issues and concepts in class.
  - Recitations will provide review and teach you the ideas you need.
  - Programming Projects (Don't freak out… yet)
  - A mid-term exam and a final exam
  - Practice homework sets
- **What do I expect of you:**
  - Come to class
  - Read the textbook (Listening to me is not good enough)
  - Work through the problems in the textbook (not really homework… but it helps)
  - Do the projects
  - Ask questions (IMPORTANT)

Questions