EECE 2560:
Fundamentals of Engineering Algorithms

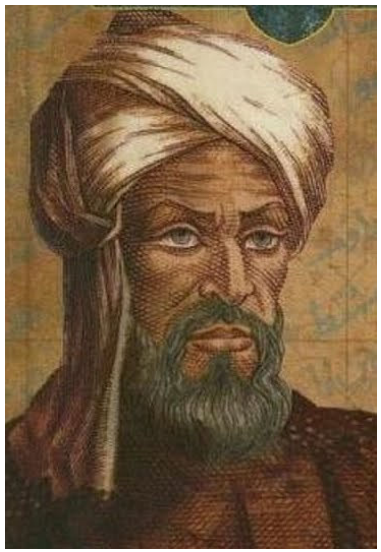
Introduction

What is an Algorithm?

- An algorithm can be viewed as a tool for solving a well-specified ***computational problem***.
  - The statement of the problem specifies the desired input/output relationship.
  - The algorithm describes a specific computational procedure for achieving that input/output relationship.
- An algorithm is said to be ***correct*** if, for every input instance, it ends with the correct output.
- An algorithm can be specified in English, in pseudo code, or as a computer program.

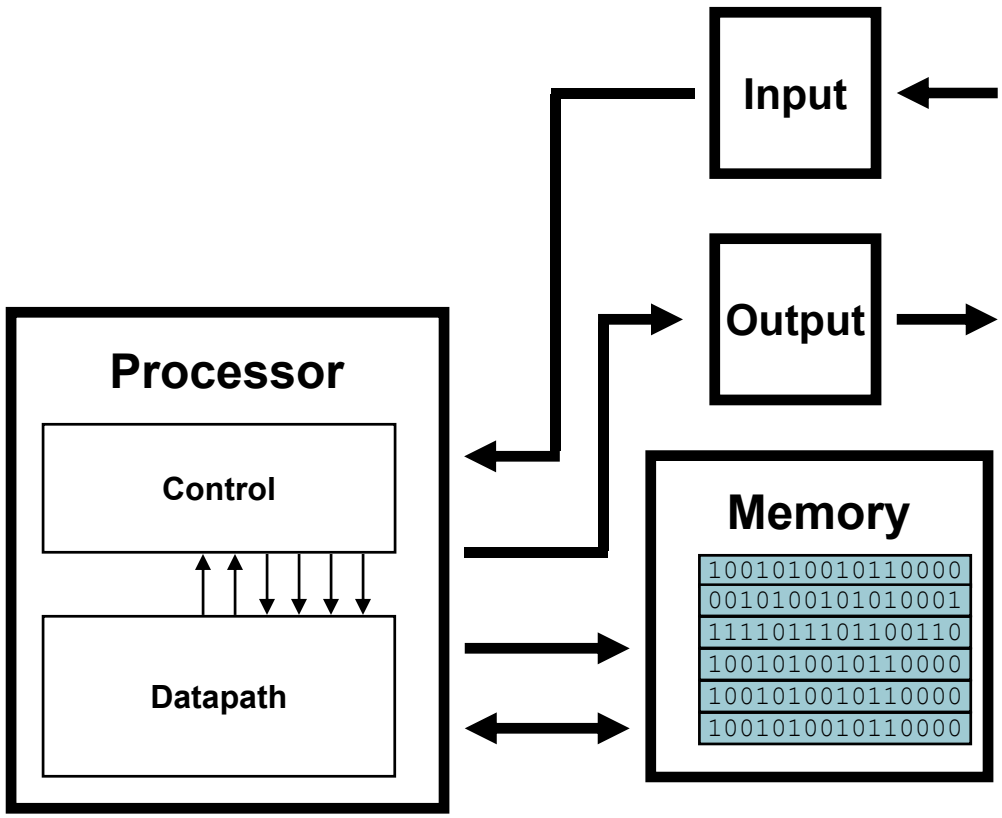
The Origin of the Word Algorithm

- The English word "algorithm" derives from the Latin form of Al-Khwarizmi's name. He developed the concept of an algorithm in mathematics.
- Al-Khwarizmi was born around 800 CE in Khwarizm, now in Uzbekistan and lived in Baghdad (Iraq).



Computer Model

- For most of this course, we shall assume a generic one processor, random-access machine (RAM) model of computation as our implementation technology
- Our algorithms will be implemented as computer programs.
- In this model, instructions are executed one after another, with no concurrent operations.



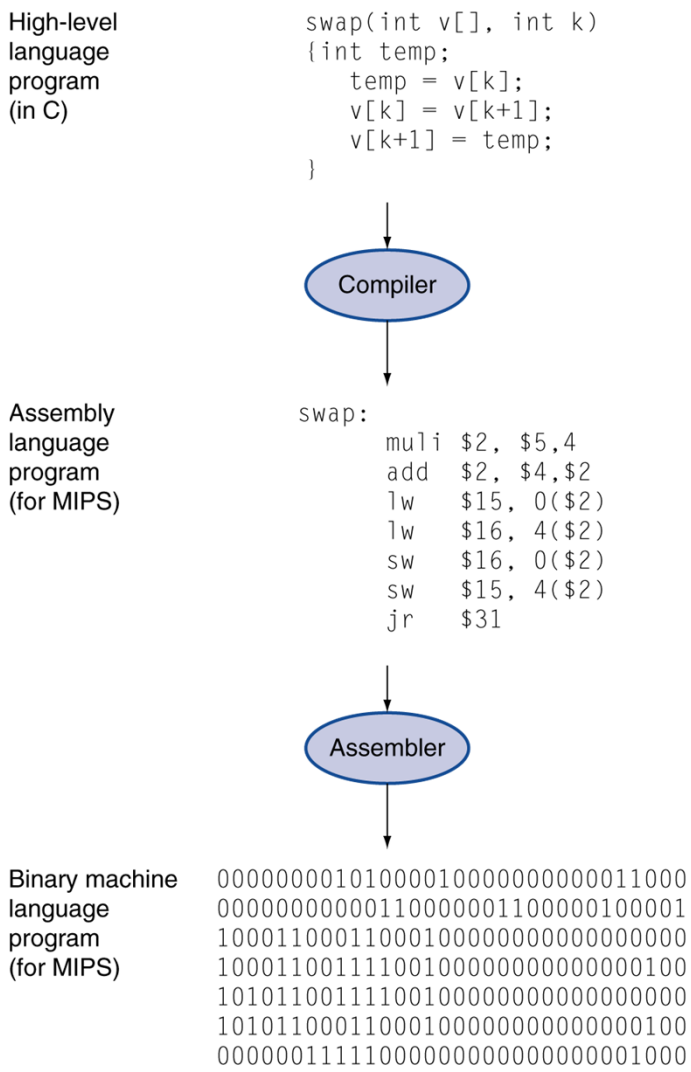
The Von Neumann Computer Model

- Same components for all kinds of computer
  - Desktop, server, embedded
- Input/output includes
  - User-interface devices
    - Display, keyboard, mouse
  - Storage devices
    - Hard disk, CD/DVD, flash
  - Network adapters
    - For communicating with other computers

John von Neumann (1903-1957) was a Hungarian-American mathematician, physicist, inventor, computer scientist

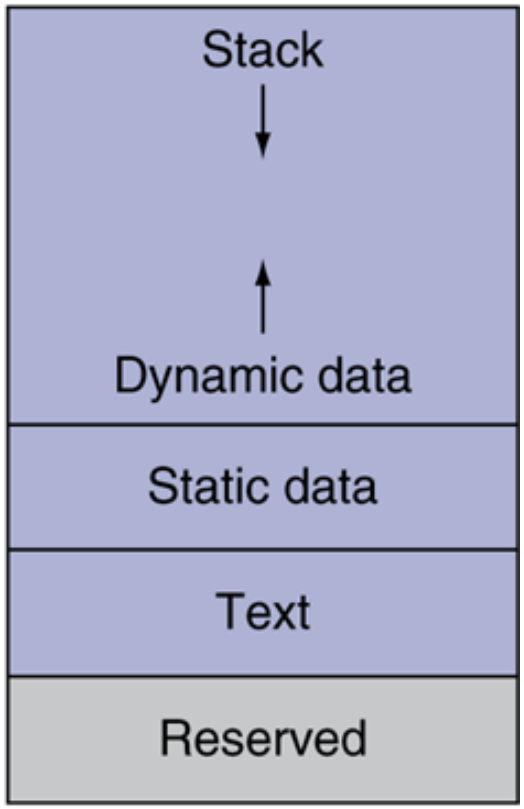
Levels of Program Code

- High-level language
  - Level of abstraction closer to problem domain
  - Provides for productivity and portability
- Assembly language
  - Textual representation of instructions
- Hardware representation
  - Binary digits (bits)
  - Encoded instructions and data



Memory Layout

- Text: program code
- Static data: global variables
  - Example: global and static variables in C.
- Dynamic data: heap
  - Examples: **`malloc`** in C, **`new`** in C++
- Stack: local variables of functions.



Computational Problem Example

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:

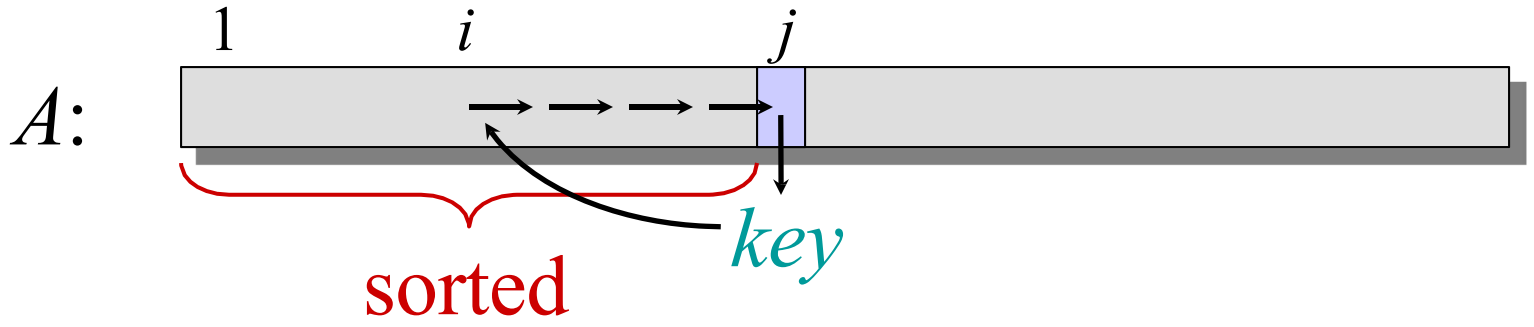
Input: 8 2 4 9 3 6

Output: 2 3 4 6 8 9

Insertion Sort Algorithm

"pseudocode"

```
INSERTION-SORT (array A , int 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
```

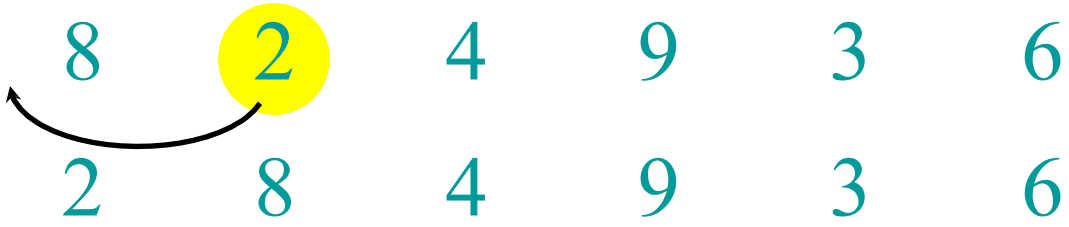


Example of insertion sort

8 2 4 9 3 6

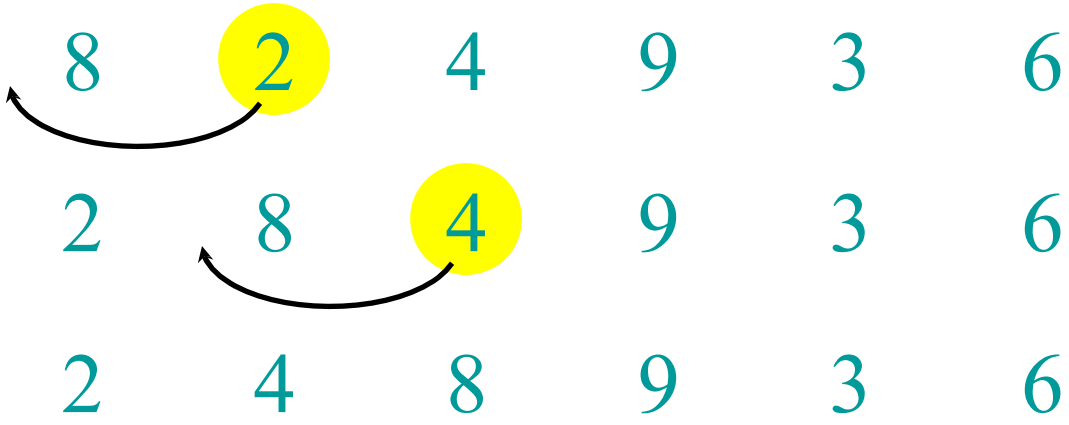
Example of insertion sort

Example of insertion sort

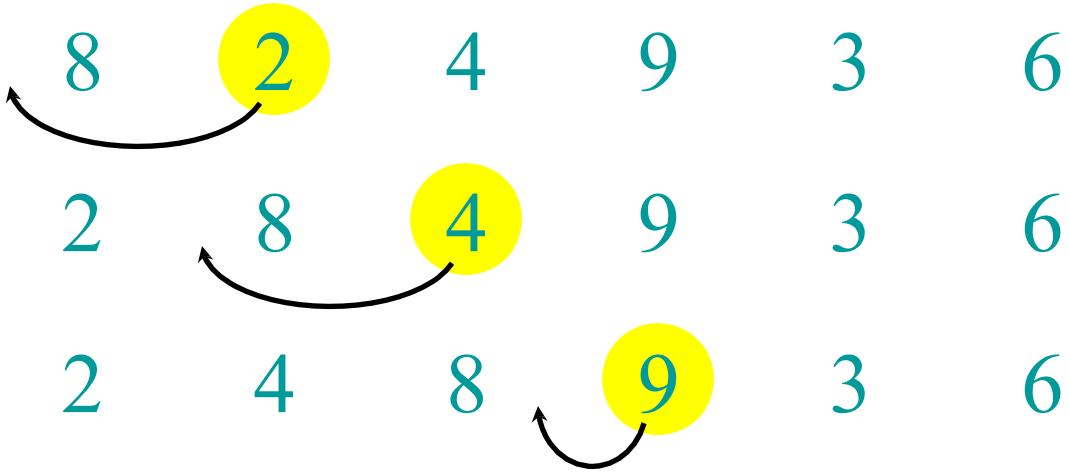


Example of insertion sort

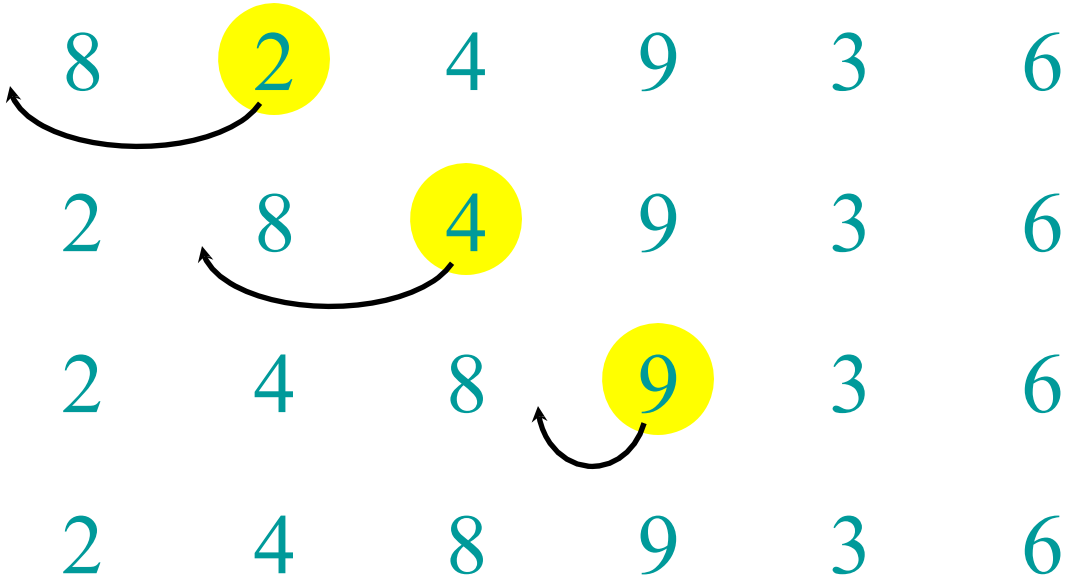
Example of insertion sort



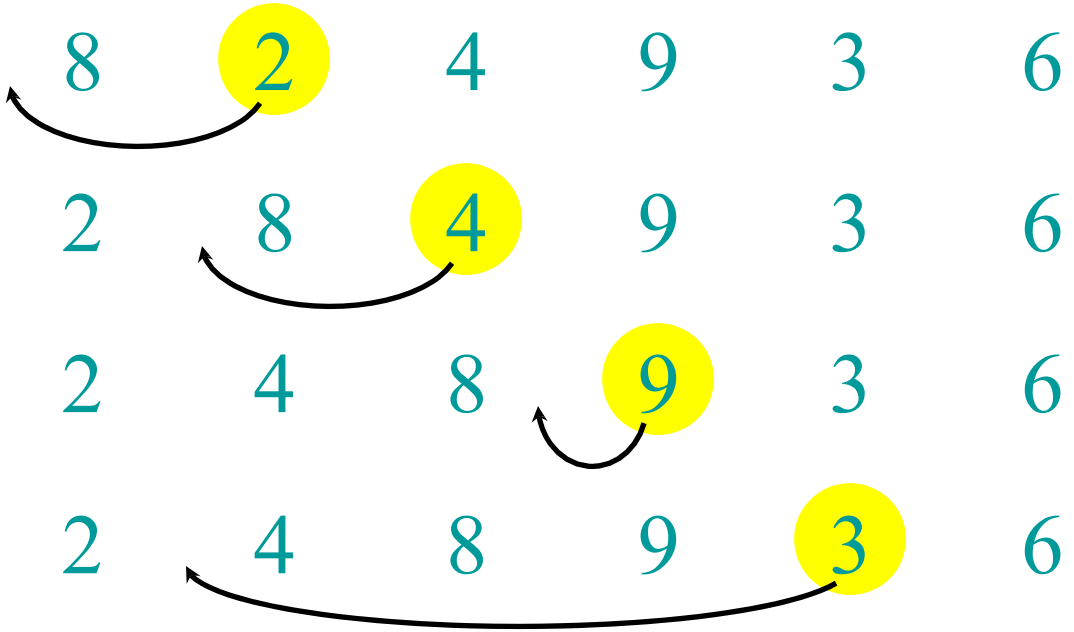
Example of insertion sort



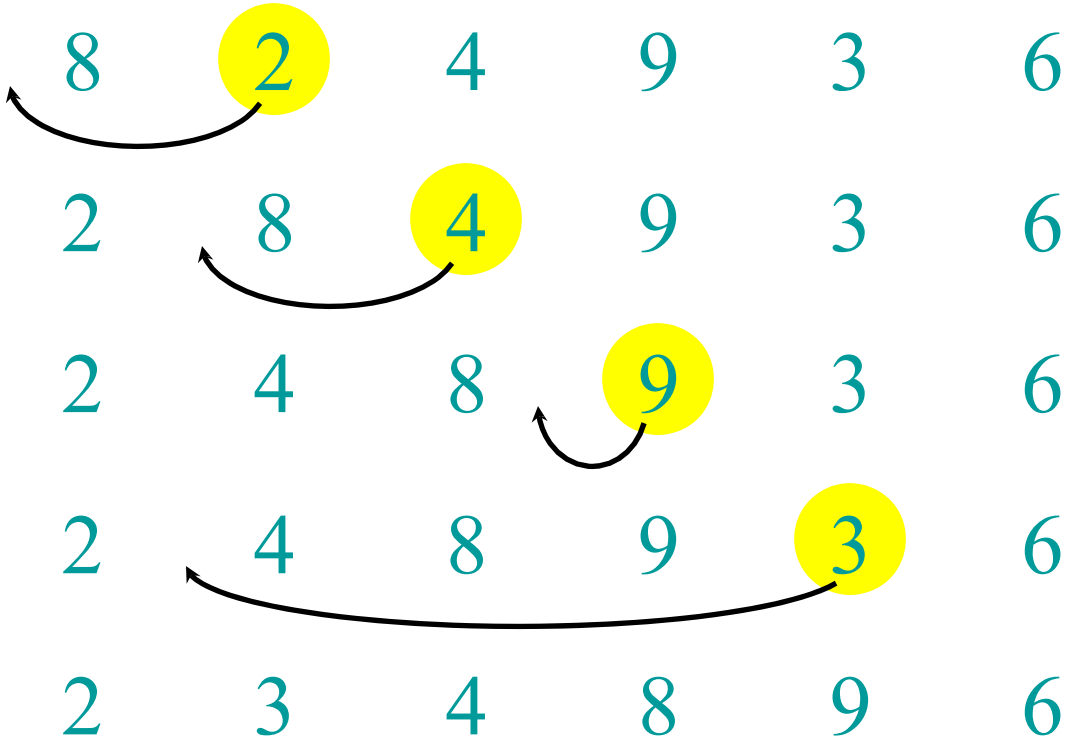
Example of insertion sort



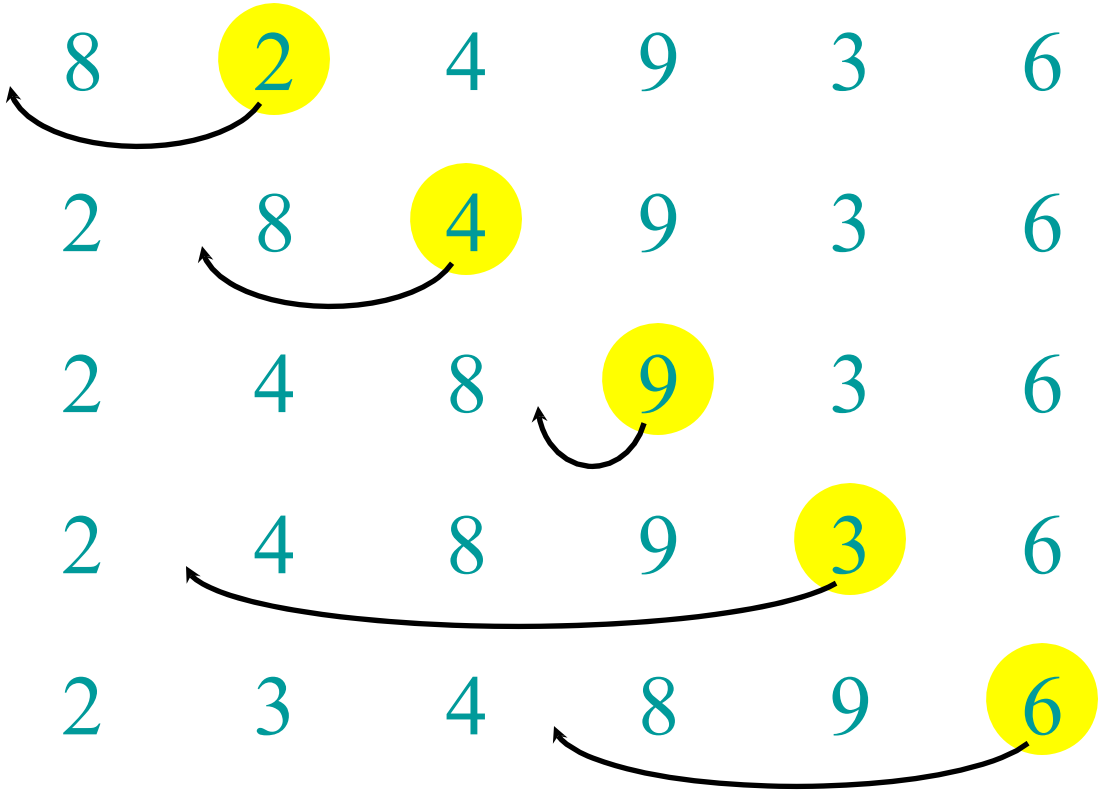
Example of insertion sort



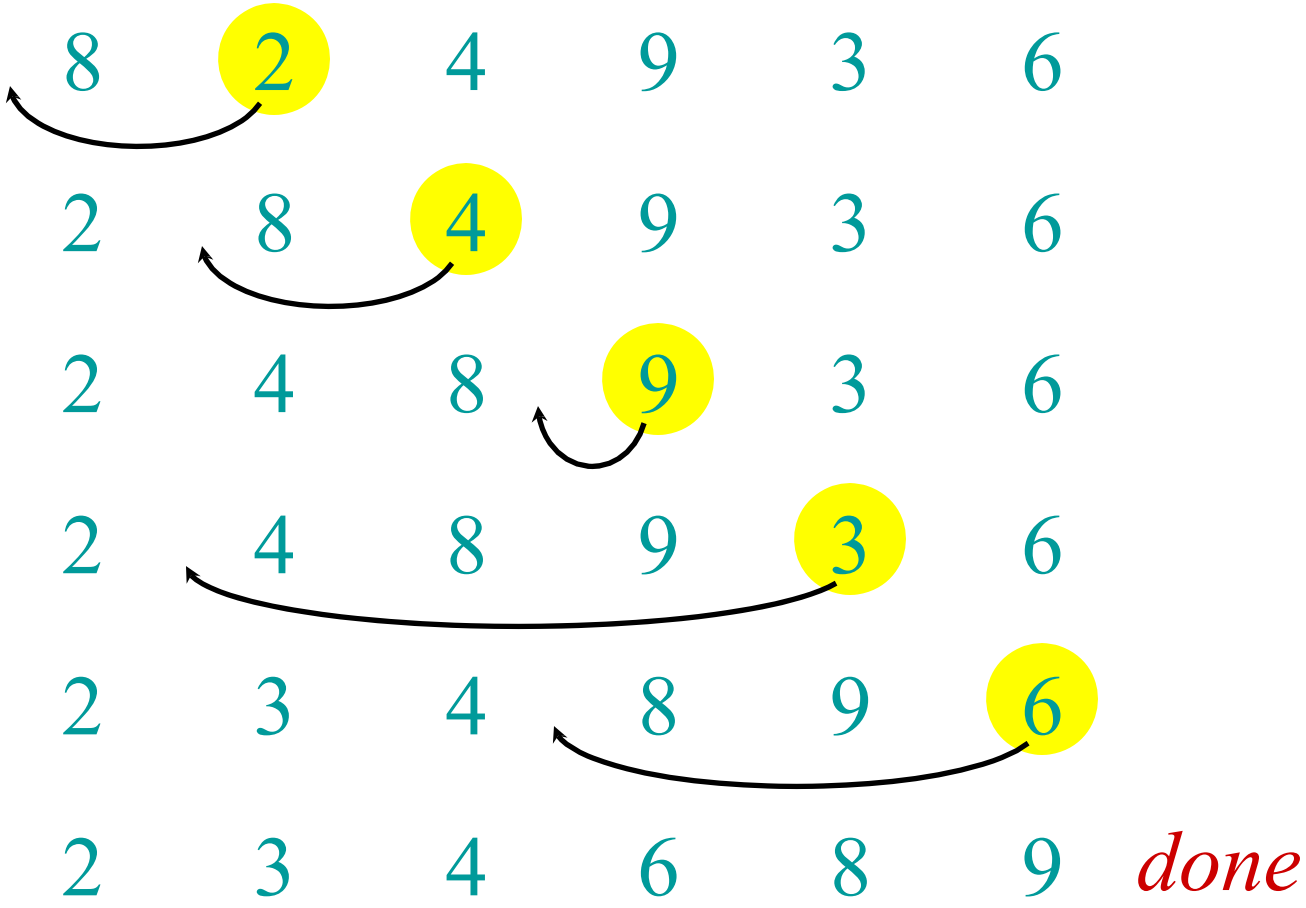
Example of insertion sort



Example of insertion sort



Example of insertion sort



Analysis of Algorithms

- Analyzing an algorithm has come to mean predicting the resources that the algorithm requires.
  - Resources such as memory, communication bandwidth, or computer hardware.
  - Most often it is computational time that we want to measure.
- Less computational time results in better computer-program performance.

Only Performance?

- What are other important features of a computer-program beside performance?
  - modularity
  - maintainability
  - robustness
  - simplicity (for programmers)
  - extensibility
  - reliability
  - memory space requirement
  - locality (for cache memory)

Performance Factors

- Algorithm
  - Determines number of operations executed
- Programming language, compiler, architecture
  - Determine number of machine instructions executed per operation
- Processor and memory system
  - Determine how fast instructions are executed
- I/O system (including OS)
  - Determines how fast I/O operations are executed

Data Structures

- A ***data structure*** is a way to store and organize data in order to facilitate access and modifications.
- No single data structure works well for all purposes, and so it is important to know the strengths and limitations of several of them.
- We will cover different data structures in this course.

Reasons to Study Algorithms

- Studying algorithms will allow you to:
  - Find an efficient solution to a new computational problem.
  - Compare different available solutions to a problem.
  - Understand different techniques used in many computing systems (e.g., operating systems, computer networks, computer architecture).
  - Finally studying algorithms will allow you to answer many interviews questions.