# COMP 102x
# Introduction to Computing with Java
# Lecture One

T.C. Pong

Department of Computer Science & Engineering

HKUST

# COMP102x Part 1 & 2

Part 1 programming fundamentals:

- Problem solving
- Primitive data types and arithmetic expressions
- Object-oriented programming basics
- Branching and Loops
- Arrays

Part 2 more advanced topics:

- String manipulation
- File I/O
- Simple event-driven programming
- Recursion
- Abstract data types

# Learning Objectives

- Take a “real-life” problem and abstract out the pertinent aspects necessary to solve it
- Formulate formal solutions to well defined problems using the logic of a programming language

# What is a well defined problem?

- A well-defined problem means that
  - a solution exists for the problem
  - the solution can be found in a finite number of steps
- Examples:
  - 1 x 2 + 3   = 5
  - 1 + 2 x 3

$$\begin{tikzcd}
 & 1 + (2 x 3) = 7 \\
1 + 2 x 3 \arrow[ur] \arrow[dr] & \\
 & (1 + 2) x 3 = 9
\end{tikzcd}$$

# Learning Objectives

- Take a “real-life” problem and abstract out the pertinent aspects necessary to solve it
- Formulate formal solutions to well defined problems using the logic of a programming language
- Implement formal solutions using Java in an integrated development environment
- Have an understanding of basics of data abstraction using the object-oriented framework

# Data Abstraction

**Data Abstraction**: To separate what can be done with the data from how it is represented.

For example, numbers can be represented in many ways:

- Arabic numerals:
  0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Roman numerals:
  I, II, III, IV, V, VI, VII, VIII, IX, X
- Chinese numerals:
  〇, 一, 二, 三, 四, 五, 六, 七, 八, 九, 十
- Numbers in computers:
  0, 1

# Computer Science

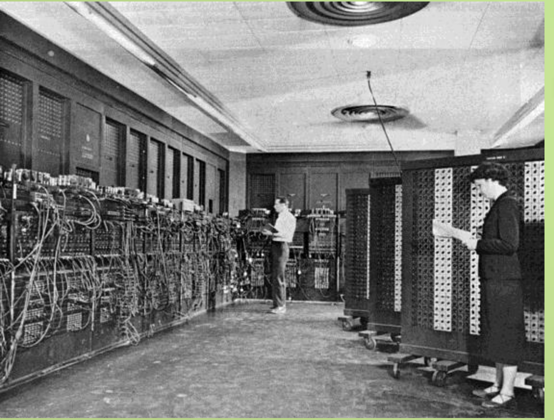
- Computer Science is a discipline that studies the theory, design and applications of computational systems.

- Computational systems
  - Hardware aspect
    - Designing and building computer systems in the form of physical devices that are able to execute programs
  - Software aspect
    - Studying the behavior of algorithms or computer programs to determine whether they are correct and efficient
  - Application aspect
    - Using computer systems to solve complex real-world problems

# Hardware and Software

- There are two important components in the computer.
  - Hardware component
  - Software component

# The First Computer



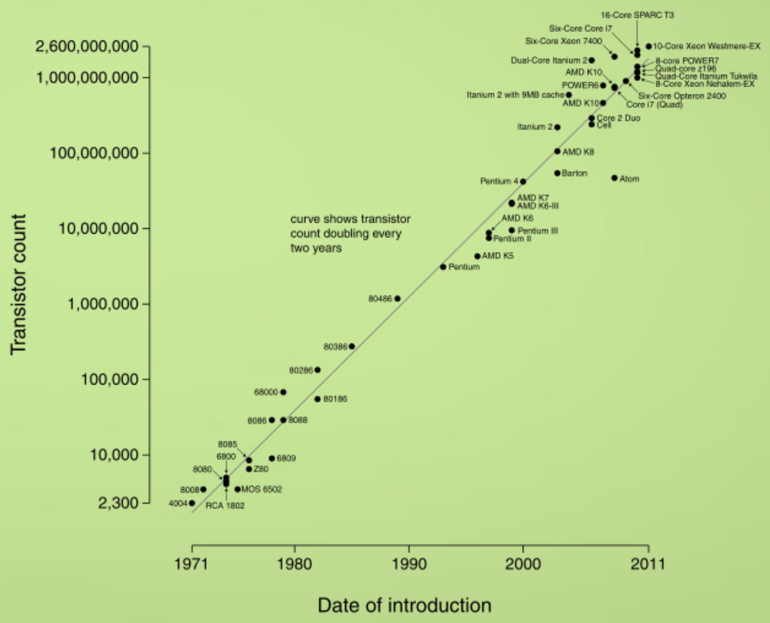
This image is from the Wikimedia Commons.
Source: U.S. Army

- The world's first all-purpose electronic computer, ENIAC, was created at U. Penn in 1946.
- It weighted over 30 tons, contained ~18,000 vacuum tubes and occupied 1800 ft$^2$ (or 180 m$^2$) of space.

# Moore’s Law

- Gordon Moore (1965): The processing power of computer chip doubles approximately every 18 months.

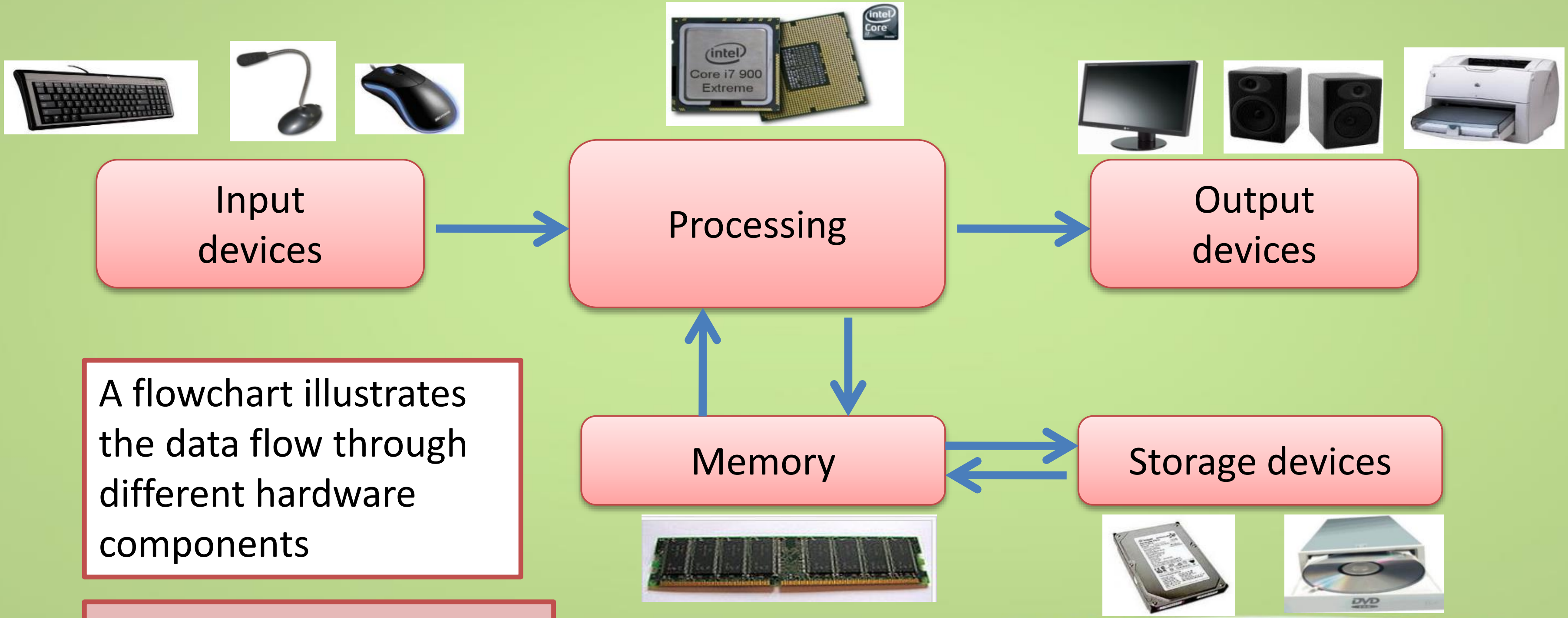
Microprocessor Transistor Counts 1971-2011 & Moore’s Law



This image is from the Wikimedia Commons. Author: Wgsimon

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

# Hardware perspective



Input devices

Processing

Output devices

Memory

Storage devices

A flowchart illustrates the data flow through different hardware components

Stored program computer

# Central Processing Unit (CPU)

Processing

- The central processing unit (CPU) is the brain of a computer
- CPUs may appear anywhere
  - Desktop computers, mobile devices such as iPhones and iPads
- It retrieves instructions from the memory and performs computation
- It consists of two sub-units:
  - Arithmetic logic unit
  - Control unit

# Arithmetic Logic Unit and Control Unit

- Arithmetic logic unit
  - Basic arithmetic operations:  + ,  - , x ,  /
  - Logical comparisons:  =  , > , <

- Control unit
  - Controlling access to the main storage
  - Controlling the sequence in which instructions are executed
  - Regulating the timing of all operations carried out within the CPU
  - Sending and receiving control signals to and from peripheral devices
  - Controlling data flow between the ALU and main memory

# Memory

Memory

- Registers: fast memory for data that the processor is working on
- Main memory: random access memory (RAM) holds the program instructions and the data for the program.
- Cache memory: stores frequently used data

# Input devices

- Get information from users
- Examples
  - Keyboard
  - Mouse
  - Microphone
- More examples?
  - You are not required to restrict your imagination on personal computers. How about the mobile phones, handheld game consoles?

Input devices

# Output devices

- Sends information from computers to users
- Examples
  - Monitor
  - Speakers
  - Printer
- More examples?

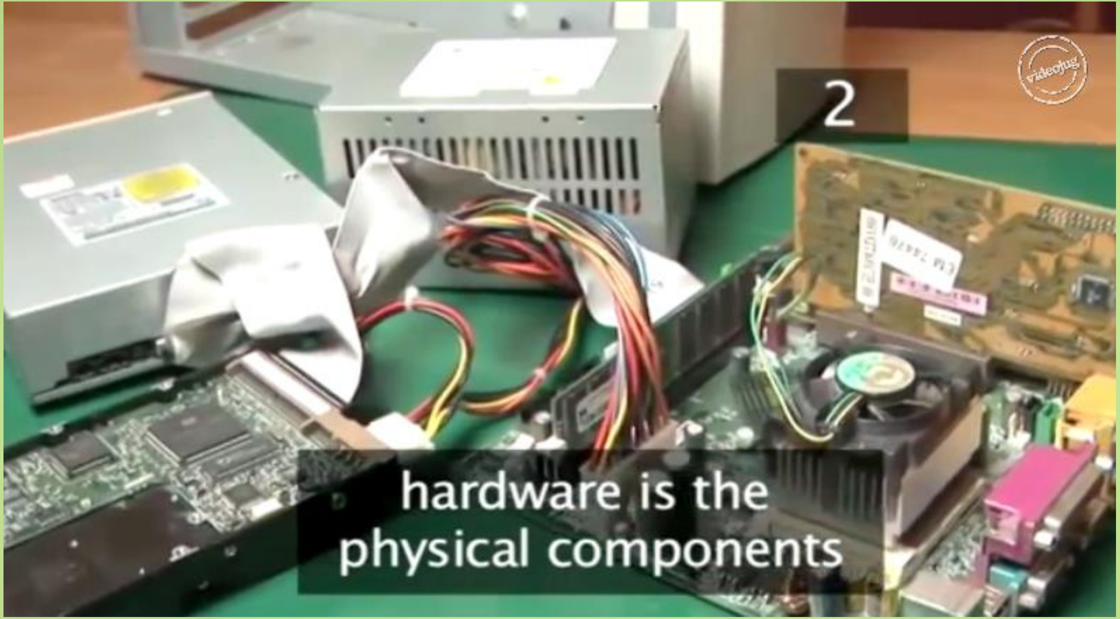
Output devices

# Storage devices

- Storage devices
  - Data will not disappear when the power is off
  - Data and programs are moved from storage devices to memory when the computer needs them
  - Examples:
    - Disk drive
    - Optical drive

Storage devices

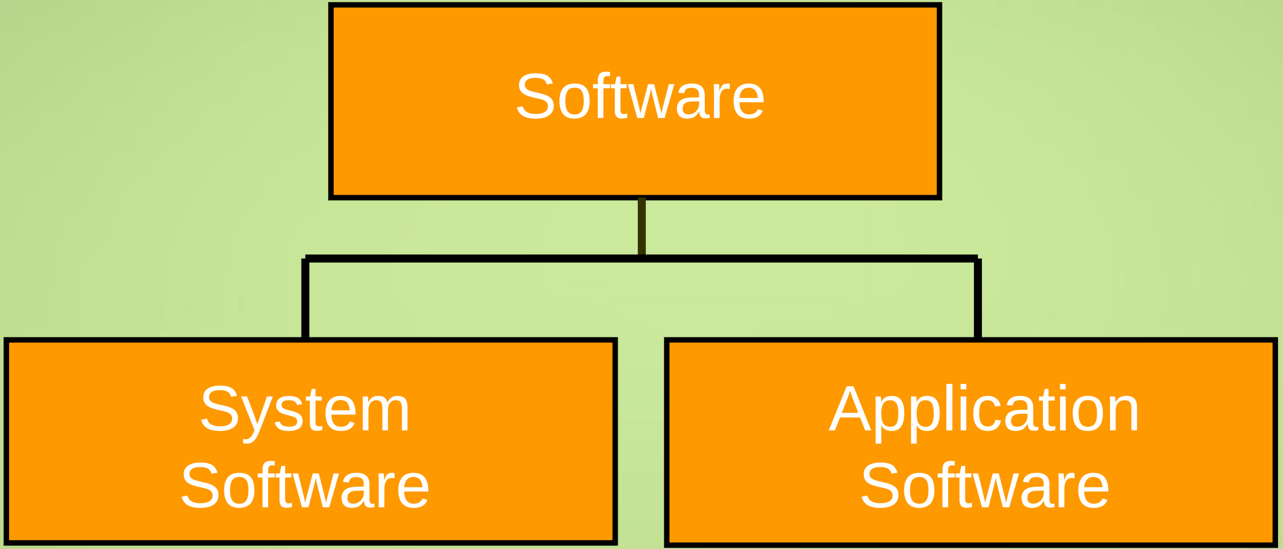
# Video: Additional information about computer hardware



http://youtu.be/yRmPTbGBqVI

Duration: 5:28

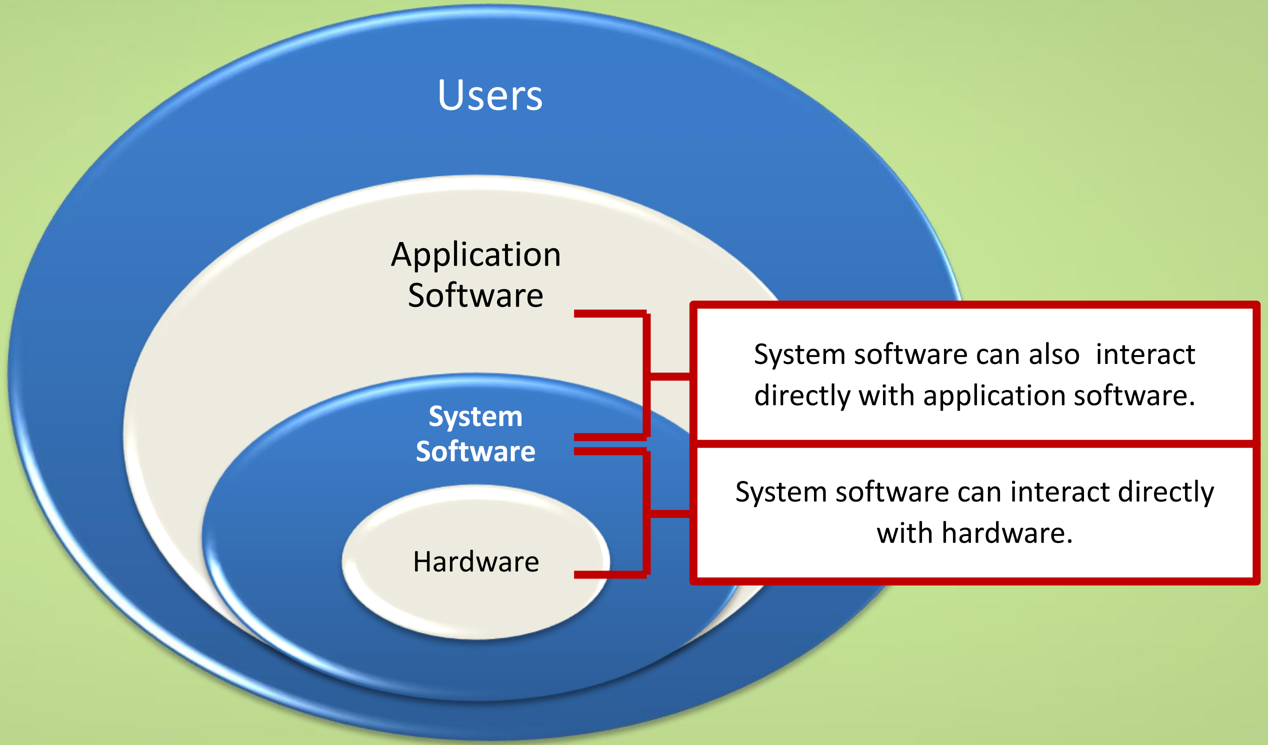
# Software Hierarchy



# Two Main Types of Software

- Application Software
  - Programs designed to perform specific tasks and are easy to use
- System Software
  - Programs that support the execution and development of other programs
  - Two major types
    - Operating systems: Window8, Unix
    - Translation systems (compilers & linkers): MS Visual C++, javac

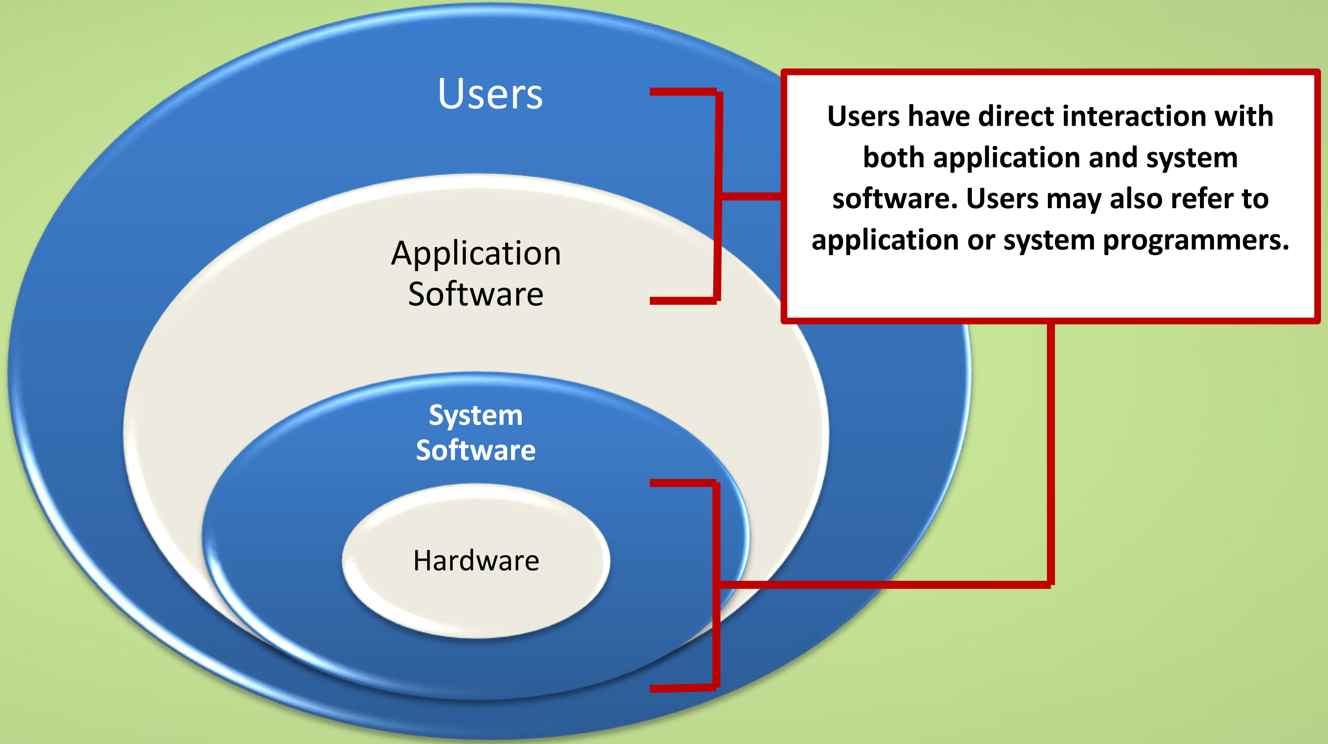
# Interface between Hardware & Software



System software can also interact directly with application software.

System software can interact directly with hardware.

# Interface between Hardware & Software



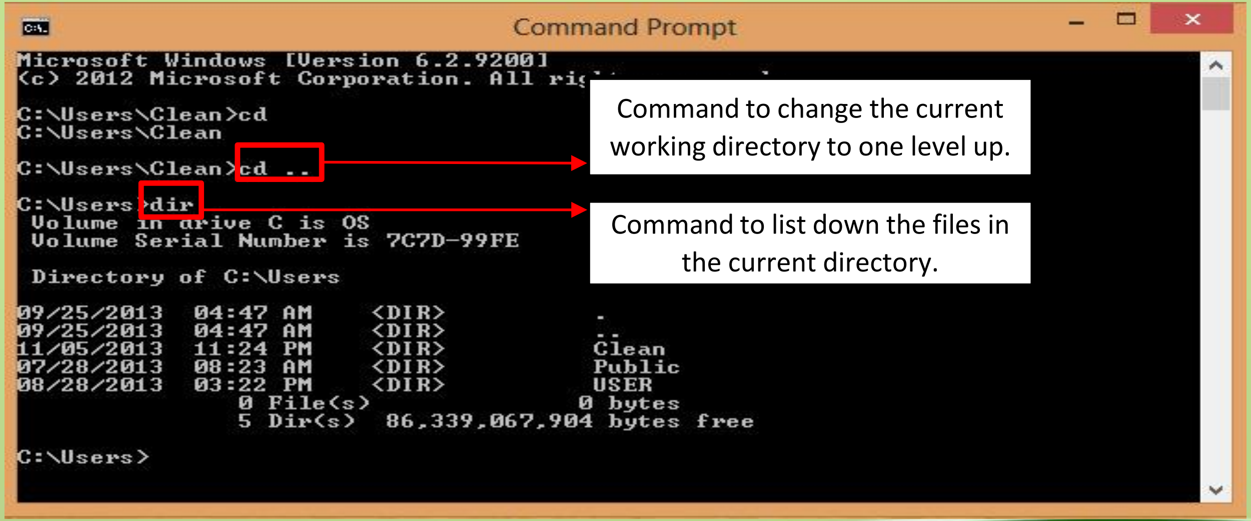
Users have direct interaction with both application and system software. Users may also refer to application or system programmers.

# User Interface

- In general, a user interface is the interaction between a user and a computer.
- A part of an OS is dedicated to providing a user interface between the OS and the user.
  - Example: The OS provides an interface for the user to manipulate the file system.
- Two types of user interfaces:
  - Command line interface
  - Graphical User Interface (GUI).

# Command Line Interface

- In a **command line interface**, the user types in **keywords** that are instructions to the OS.
- Example: Command prompt in PC systems



```
Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. All rig

C:\Users\Clean>cd
C:\Users\Clean

C:\Users\Clean>cd ..

C:\Users>dir
 Volume in drive C is OS
 Volume Serial Number is 7C7D-99FE

 Directory of C:\Users

09/25/2013  04:47 AM    <DIR>          .
09/25/2013  04:47 AM    <DIR>          ..
11/05/2013  11:24 PM    <DIR>          Clean
07/28/2013  08:23 AM    <DIR>          Public
08/28/2013  03:22 PM    <DIR>          USER
               0 File(s)              0 bytes
               5 Dir(s)  86,339,067,904 bytes free

C:\Users>
```

Command to change the current working directory to one level up.

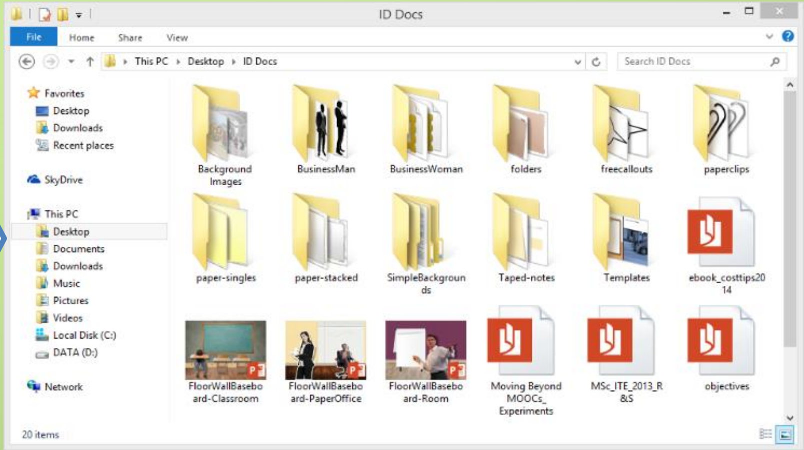
Command to list down the files in the current directory.

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

# Graphical User Interface

- In a **graphical user interface**, the computer screen displays graphical objects such as **icons**, **menus**, and **windows**.
- The user has a **pointing device** to activate functions related to the objects.

This file system is represented graphically and the commands to show, move or delete files and folders are triggered by mouse click actions.



# Application Software

- Application software has made using computers easy and popular.
- Application running on computer is efficient and reliable.
- Business applications or home applications.
- Improve your quality of life, hopefully.
  - Program that can handle your tax filing system.
  - Develop a system call "Smart Home" that can control different appliances at home.

# Common Application Software

- Productivity software
  - Microsoft Word, PowerPoint
- Communication software
  - Safari, Internet Explorer, Firefox, Chrome
- Multi-media software
  - PhotoShop, QuickTime
- Mobile apps
  - Games, communication, entertainment, navigation
- Education software
  - Blackboard, Moodle, Massive Open Online Course (MOOC) platforms

# Massive Open Online Courses

A **massive open online course (MOOC)** is a type of online course aimed at large-scale participation and open access via the Web.

**Key features:**

- **Open access** - MOOC participants do not need to be a registered student in a school to "take" a MOOC, and they are usually free.
- **Scalability** - MOOC is designed to support a large number of participants.

**Major MOOC platforms include**

- Coursera: www.coursera.org
- edX: www.edx.org

# Operating System

- Controls and manages the computing resources
- Examples:
  - Apple: Mac OS (OS X), iOS
  - Microsoft: MS DOS, Windows
  - Others: Unix, Linux, Android
- Important services that an operating system provides:
  - Manage file system
  - Interpret commands to manipulate the file system
  - Input and output on a variety of devices
  - Window management

# Programming Languages

- Low-level programming languages
  - Machine language
  - Assembly language
- High-level programming languages
  - Fortran: mainly used in scientific application
  - Cobol:  mainly used in business application
  - Basic: one of easiest language to use
  - C:  predecessor of C++
  - Python: support multiple programming paradigms
  - Java: object-oriented language

# Software Development Flow

- Major activities in the software development cycle:
  - Editing the source code
  - Compiling (creates **`.obj`** file)
  - Linking with compiled files (creates **`.exe`** file)
    - Object files
    - Library modules
  - Loading and executing
  - Testing the program

# Integrated Development Environment (IDE)

- An Integrated Development Environment is an application software that provides interactive tools for programmers to streamline the software development cycle.
- Common components in an IDE:
  - Editor
  - Compiler
  - Linker
  - Loader
  - Debugger
    - Syntax and semantic errors

**• To feed her dog food**

**• To move the mouse on my desk**

# Problem Solving Process

- Define and analyze the problem.
- Develop a solution.
- Write down the solution steps in detail.
- Test and evaluate the solution, revise if necessary.
- Document and maintain the solution.

# Example

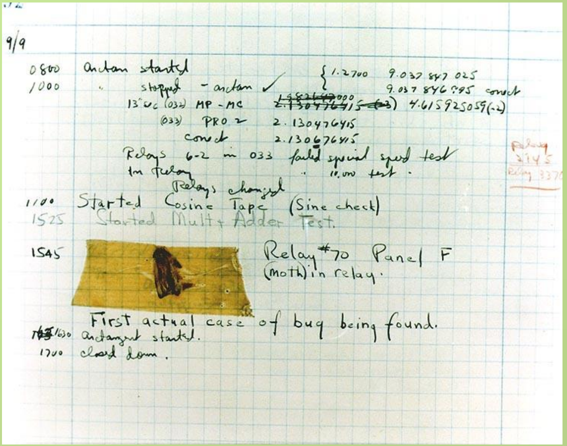
- Problem Definition:
  - To find the best way to travel from Hong Kong to London for your client.
- Problem Analysis:
  - The exact locations in Hong Kong and London
  - Do you want shortest distance, fastest time or cheapest cost?
- Preliminary Design:
  - Consider all possible routes and modes of transportation
- Refined Design:
  - Analysis all possible routes and modes of transportation
  - Select the route that meets our goal (fastest and cheapest)

# Example (Cont.)

- Testing and evaluating:
  - Check the on-time track record of flights
  - Check with those who have tried similar route before
- Documentation:
  - Document the itinerary for your client to follow
- Maintenance:
  - Collect feedback on the trip from your client
  - Revise the solution if your client was not happy with the experience

# Programming as a Problem Solving Process

- Problem Formulation: define and analyze the problem.
  - What are the input & output?
  - What other information is necessary?
  - Find an appropriate representation for the problem.
- Develop an algorithm.
  - What are the steps for solving the problem?
  - A sequence of precise steps for carrying out a function.
- Implement a program.
- Compile, test, and debug the program.
- Document and maintain the program.



9/9

0800 Anctan started
1000 " stopped - anctan ✓ {1.2700 9.037 847 025
9.037 846 995 conect
13" uc (032) MP - MC ~~1.982147000~~ ~~2.130476415~~ (~~-3~~) 4.615925059(-2)
(033) PRO 2 2.130476415
conect 2.130676415
Relays 6-2 in 033 failed special speed test
In relay " 10,000 test .
Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Mult + Adder Test.
1545 Relay #70 Panel F
(moth) in relay.
First actual case of bug being found.
~~1600~~ 1630 Antangent started.
1700 closed down.

Relay 2145
Relay 3370

This image is from the Wikimedia Commons. Source: U.S. Naval Historical Center Online Library Photograph NH 96566-KN

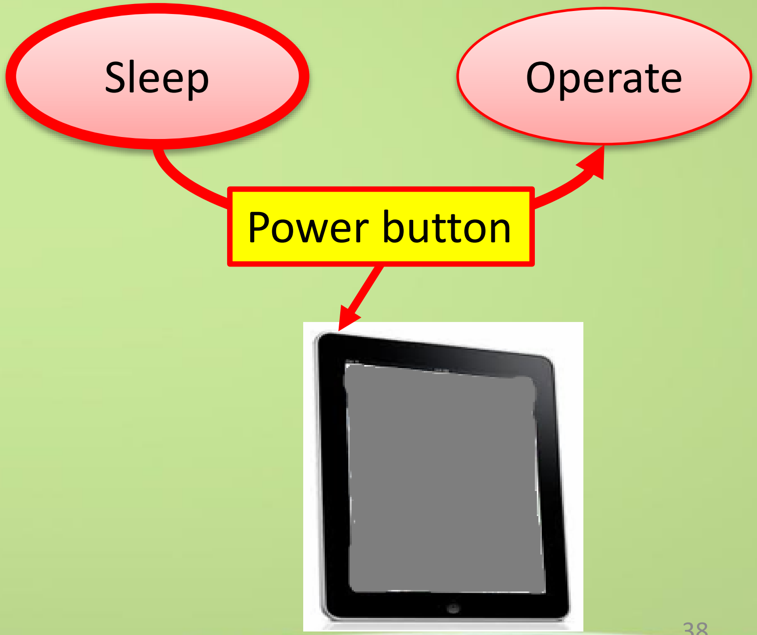
# Problem Representation

Problem formulation using a state space representation:

- A problem is represented as a set of states
- A state space is the set of all possible states, including
  - initial states
  - final states
- Two states are connected if there is an operation that can transform one state to the other

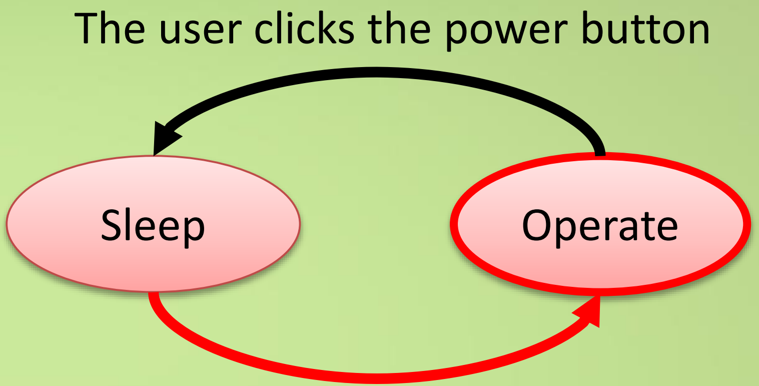
# State Space Representation

- An iPad example is used to illustrate two states and its relationship
- Initially, the iPad is in the "Sleep" state



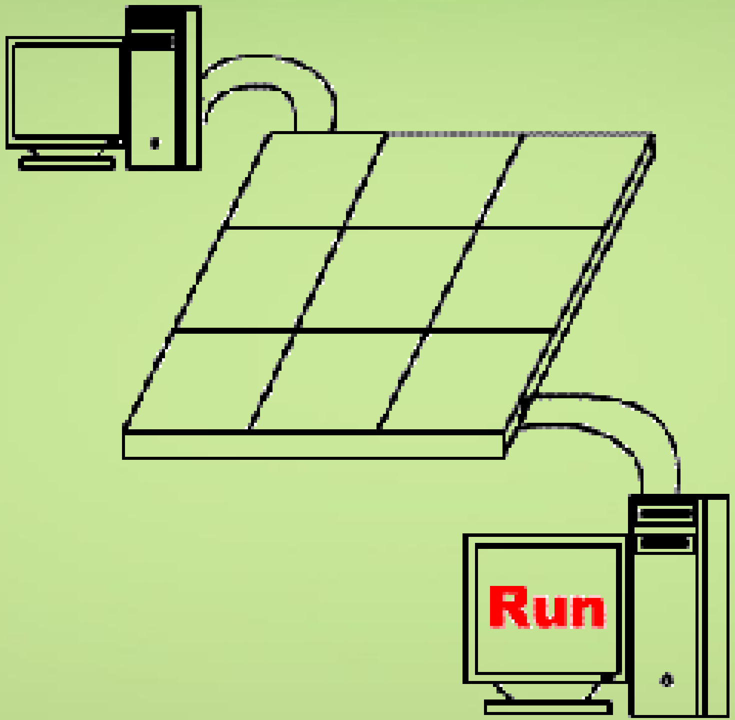
# State Space Representation

- An "event" triggers the change of state
  - The user clicks the power button
  - The iPad switches from the "Sleep" state to the "Operate" state
  - When the user clicks the power button again, it switches back to the "Sleep" state

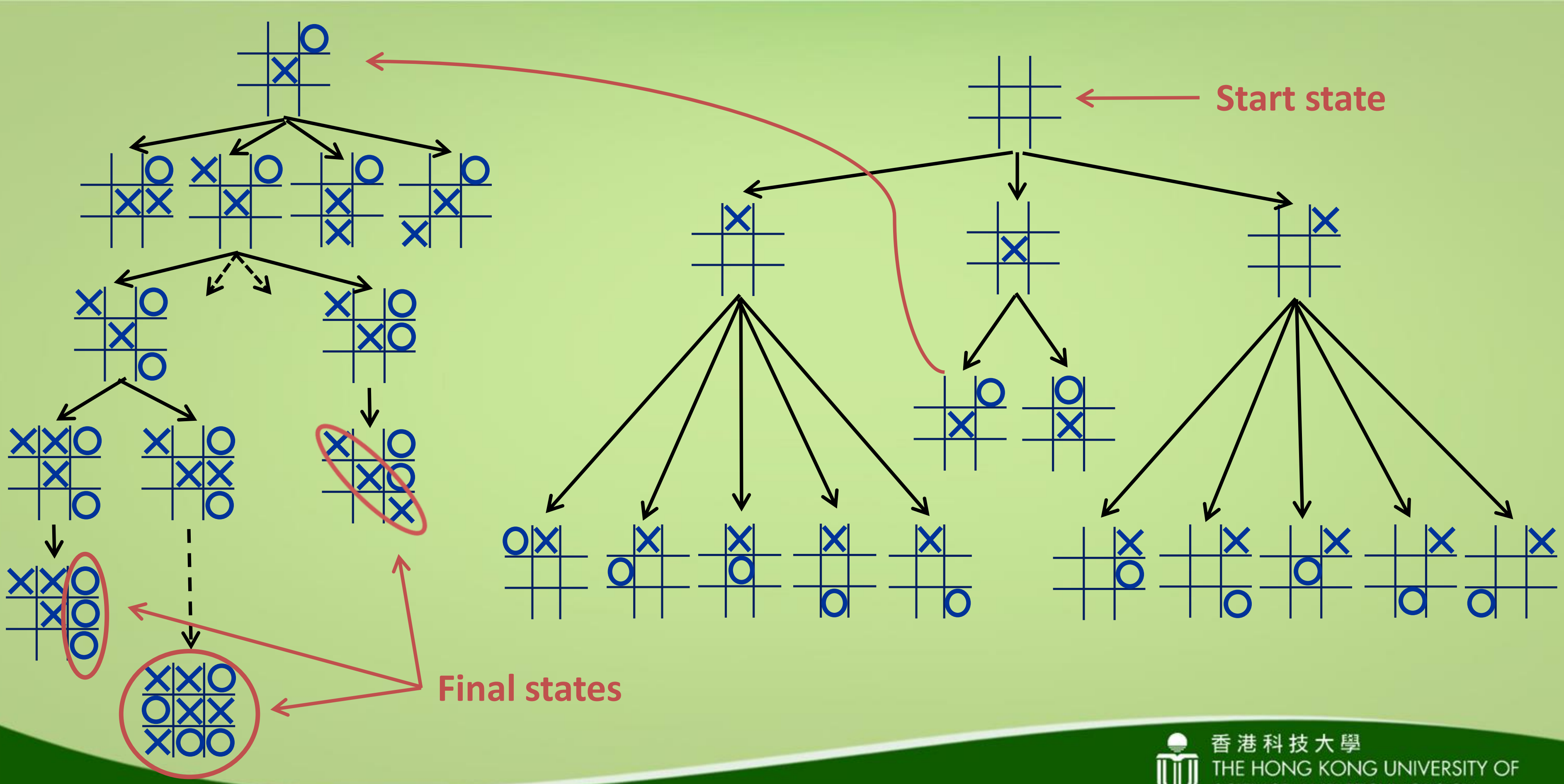


The user clicks the power button

# The game of Tic-Tac-Toe

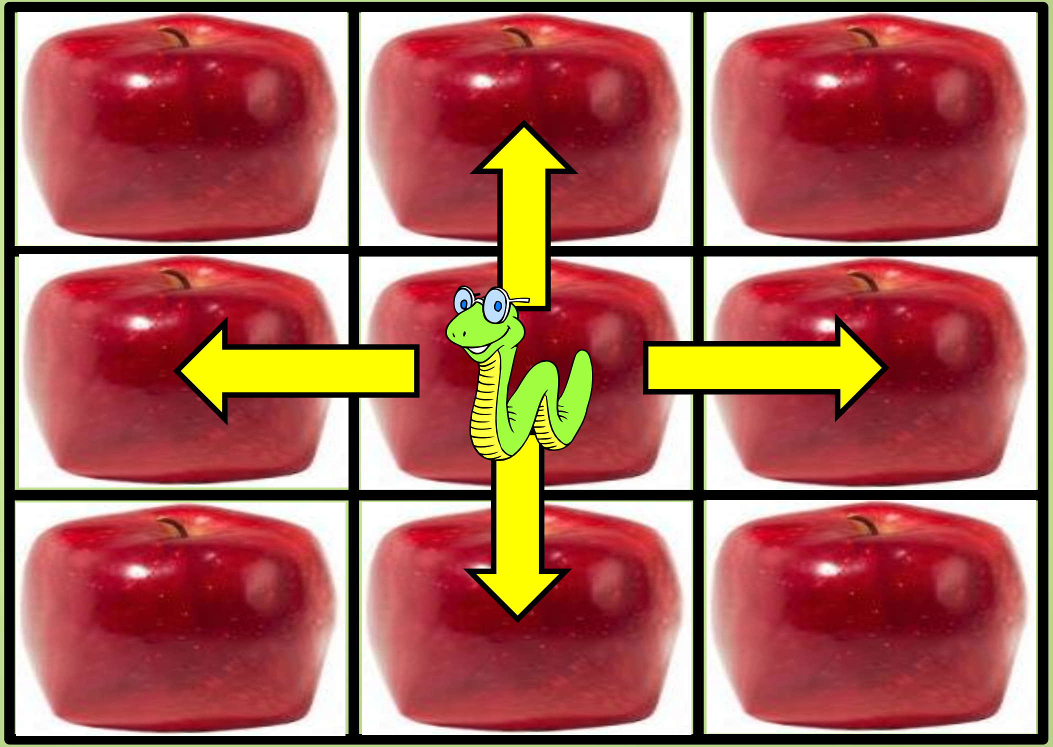


# State Space Representation for Tic-Tac-Toe



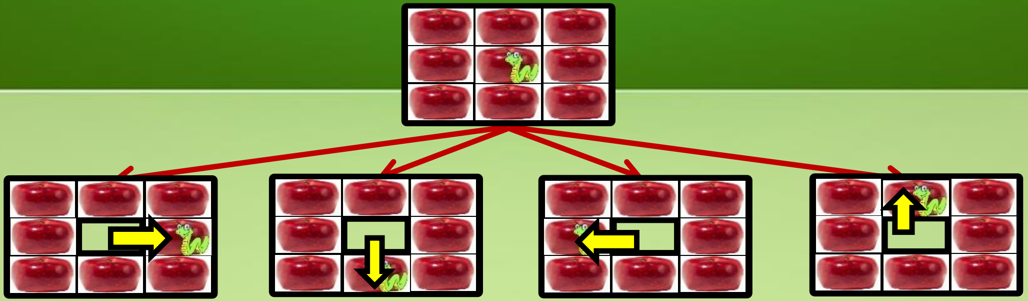
# Square Apple Problem

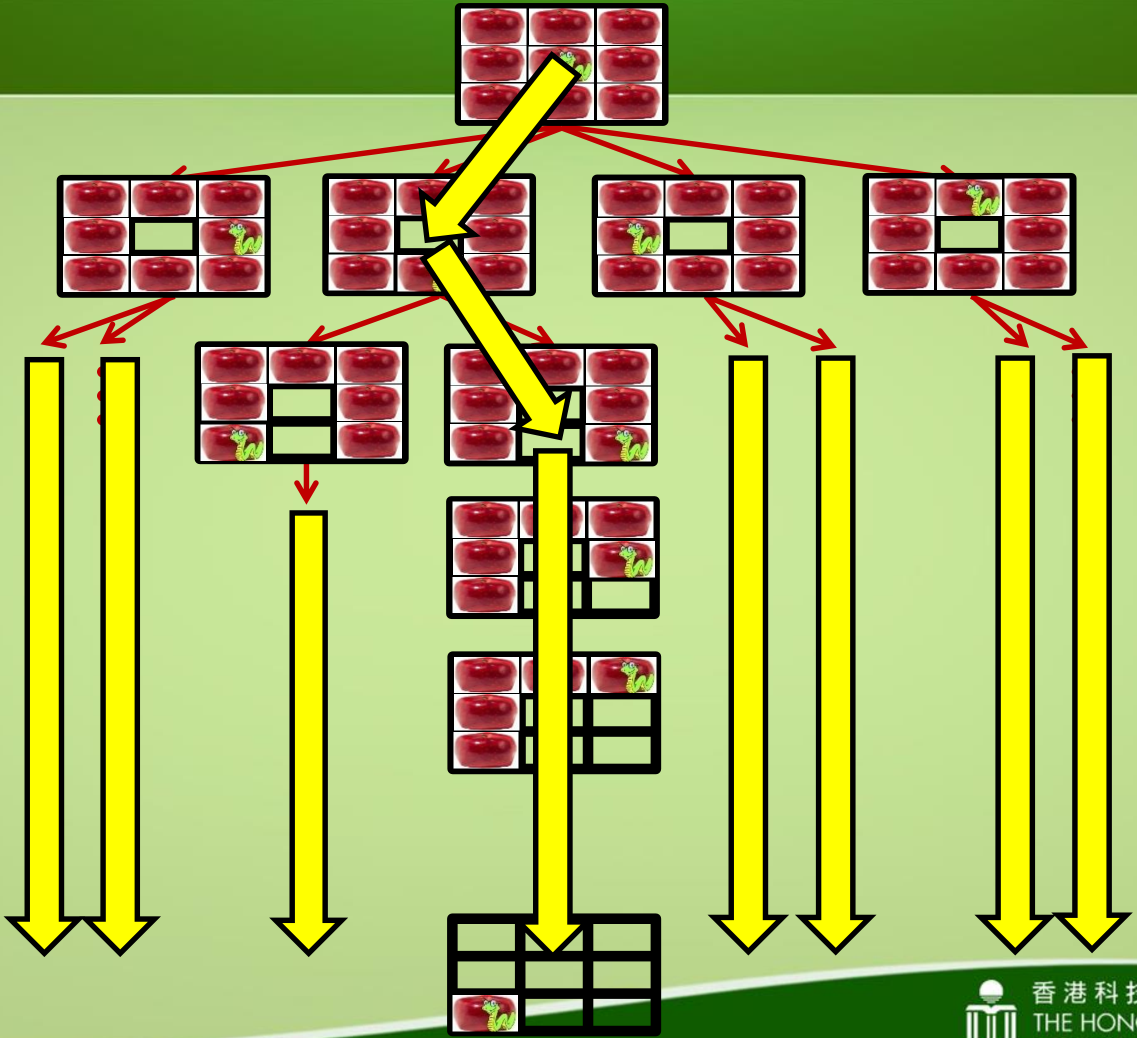
Starting from the middle cell, would it be possible for the worm to finish eating all the apples?



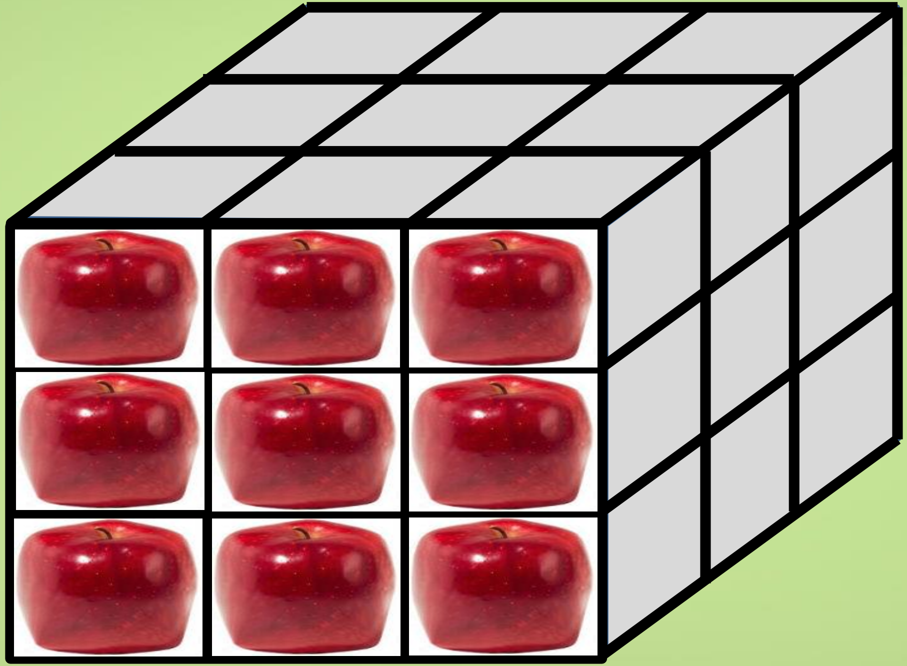
Rules:

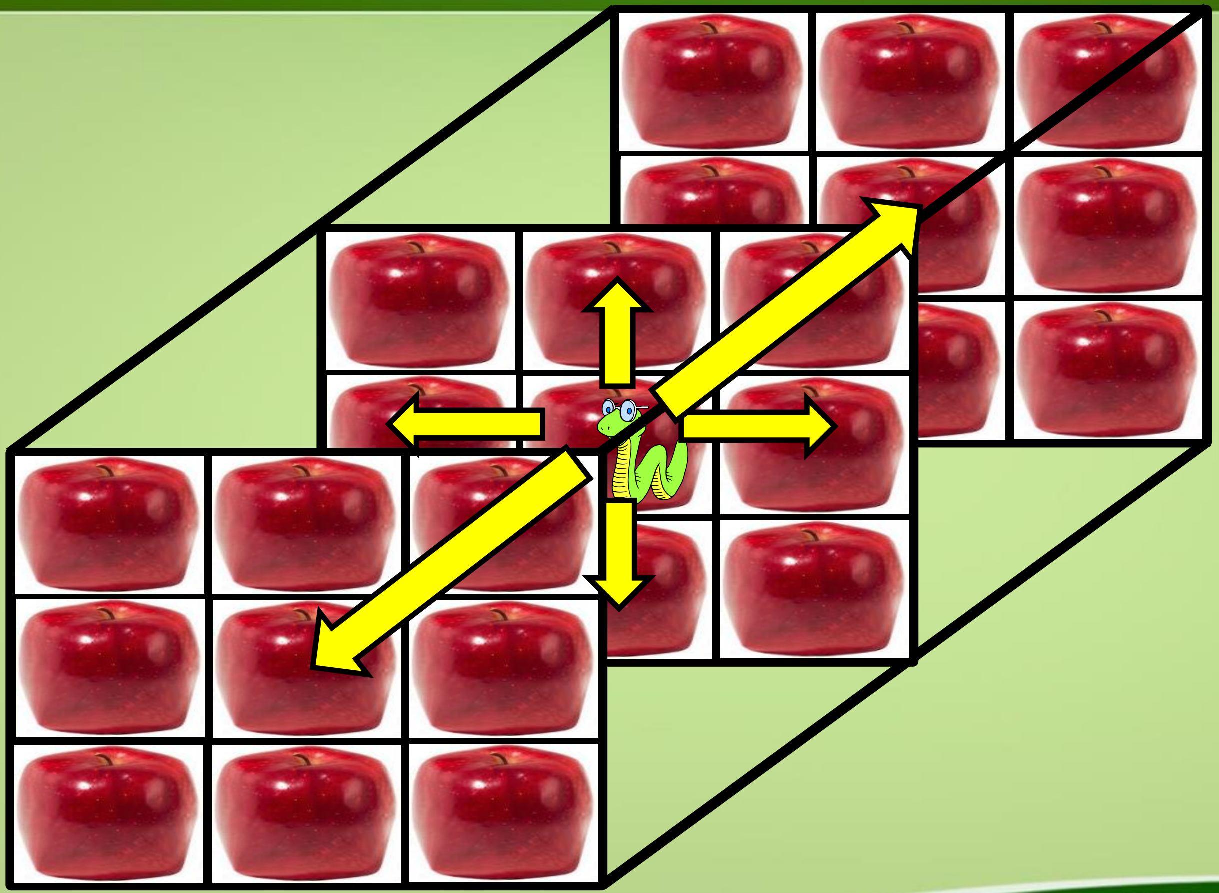
- The worm can only move into another cell that shares a common wall; and
- a cell that has not been previously visited.



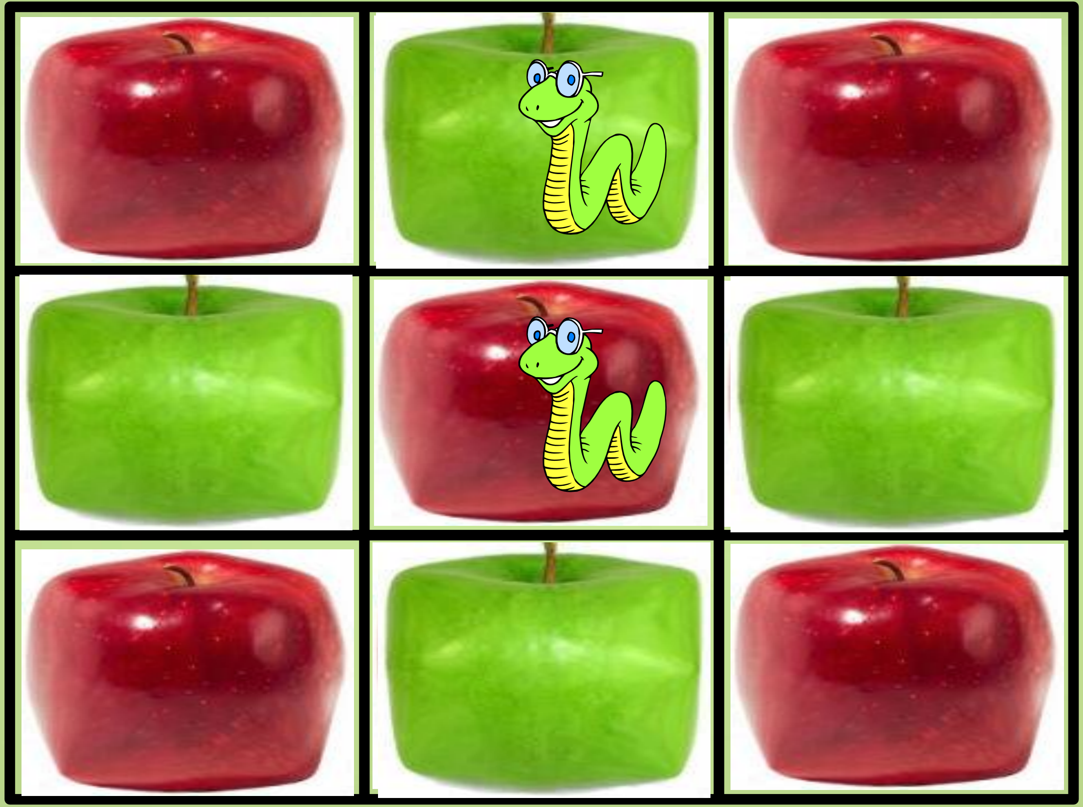


# 3D Square Apple Problem

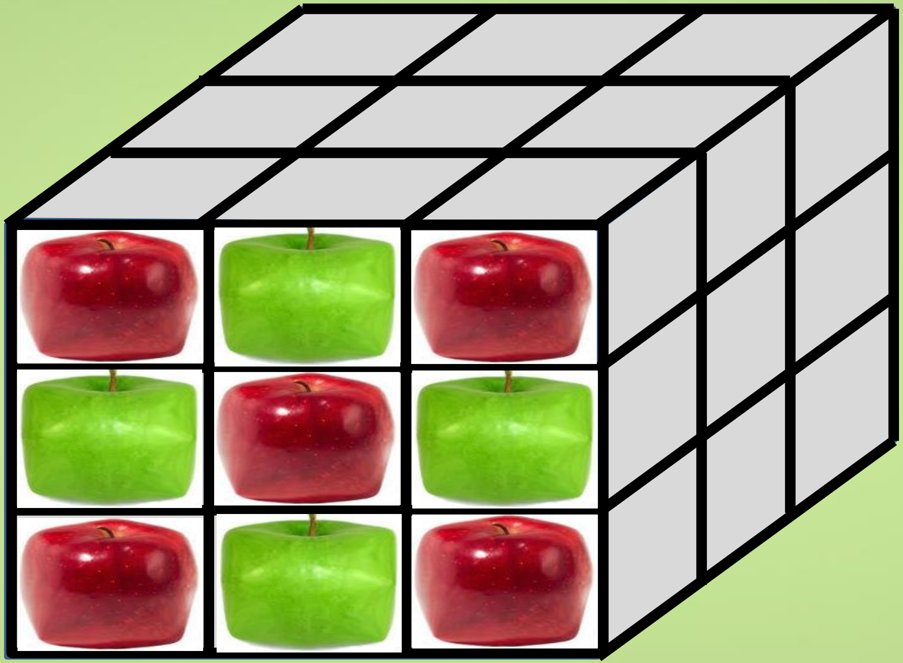


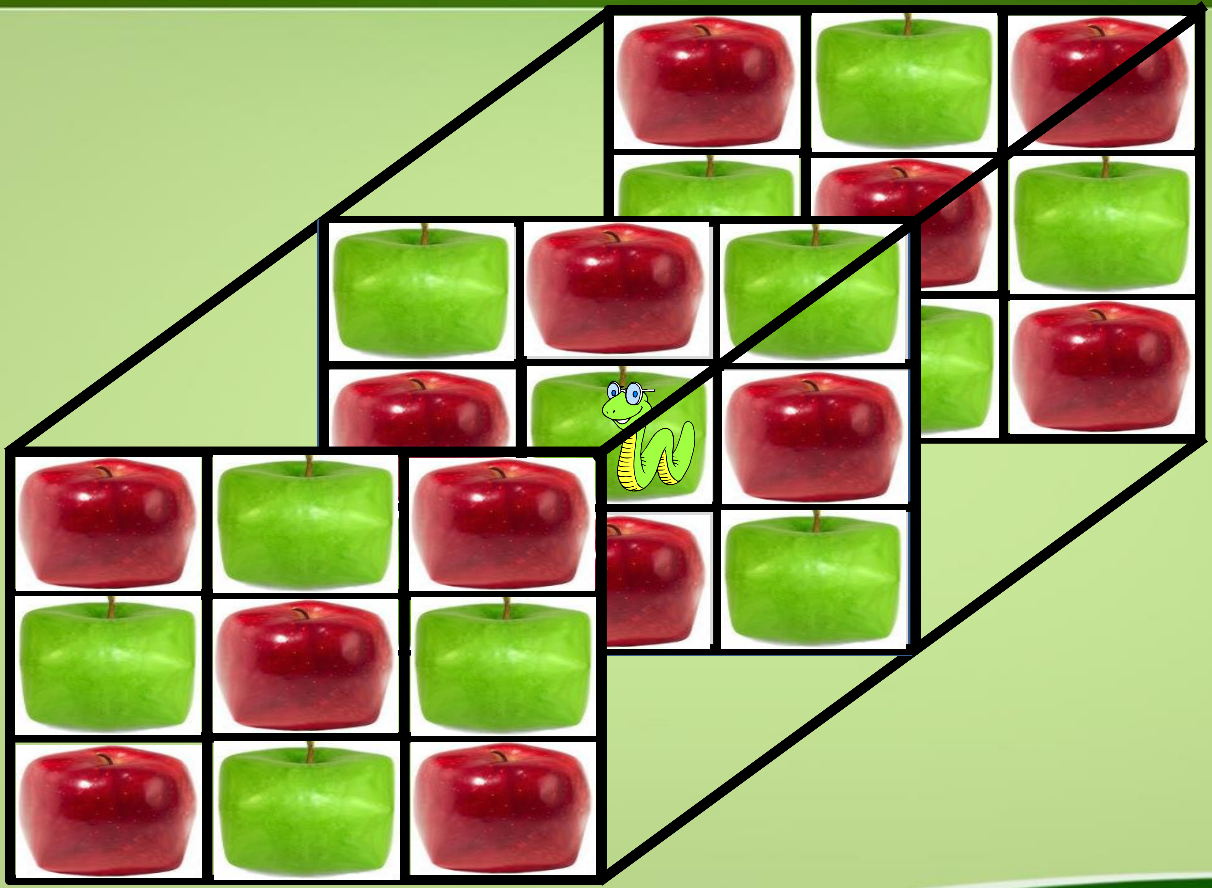


# 2D Square Apple Problem

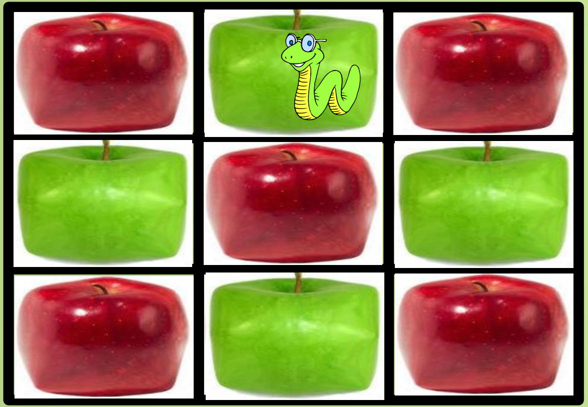


# 3D Square Apple Problem





# Importance of Problem Representation



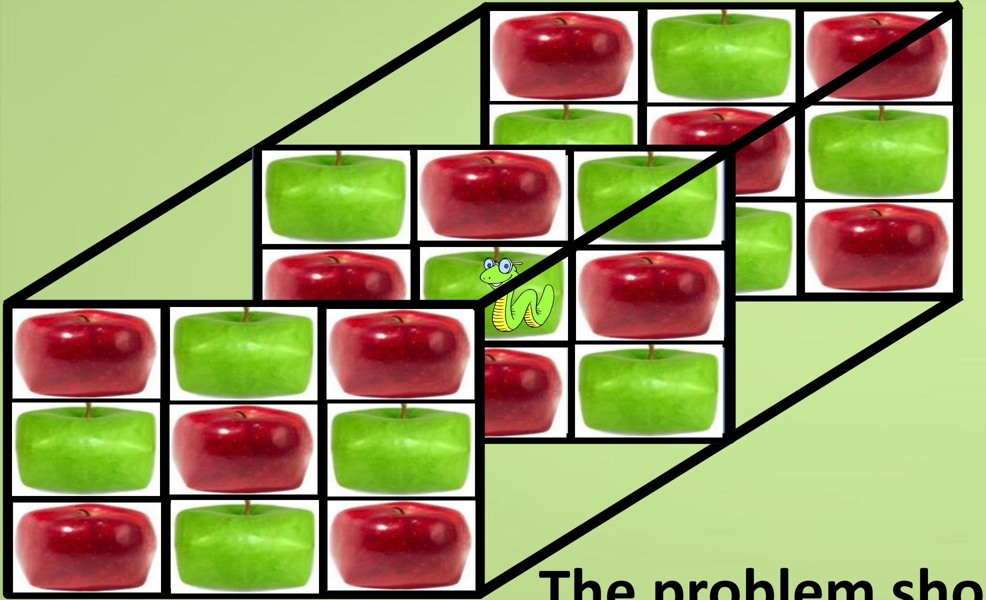
**2D square apple:**

- There are 5 red apples and 4 green apples
- Every move will be restricted to going between apples of different colors
- If one starts from a green apple, there would be no more green apple after consuming the 4th red apple

# Importance of Problem Representation

**3D square apple:**

- There are 14 red apples and 13 green apples



Using the same argument as in the 2D case, if one starts from a green apple, there would be no more green apple after finishing the 13th red apple.

**The problem shows the importance of finding the right representation before solving a problem!**

# Your First Java Program

Comments

The name main indicates start of program

Statement for printing out the greeting message

```
// a simple program sends a greeting to the world
public class HelloWorld
{
    public static void main(String[] args)
    {
        System.out.println("Hello world!”);
    }
}
```