Basics of
Data Structures and Algorithms

Jyh-Shing Roger Jang (張智星)

CSIE Dept, National Taiwan University

Programming != Coding

- Programming ➔ Building a house
  - Requirements: purpose, input/output ➔需求為何、投入多少資金、產出什麼品質的房子
  - Analysis
    - Bottom-up: small pieces to ultimate goal ➔ 把每一面牆、每一塊磚設計好，再想辦法拼起來
    - Top-down: ultimate goal to small pieces ➔ 先考慮整體的需求，在思考每一面牆、每一塊磚如何完成
  - Design: choices of data structures and algorithms ➔ 建材和工法的選定
  - Coding and refinement: actual implementation ➔ 施工
  - Verification
    - Proof (in math) ➔ 確認是否符合設計圖，例如載重度或耐震度
    - Test and debug (on machines) ➔ 工地現場的牢固度測試、監工等

From Coding to Programming

- Comparison of DSA with “Intro to C”

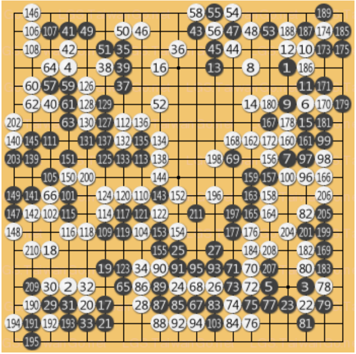
	Intro to C	DSA
Requirements	*	*
Analysis & Design	*	***
Coding	***	***
Proof in Math	0	***
Test & Debugging	**	***

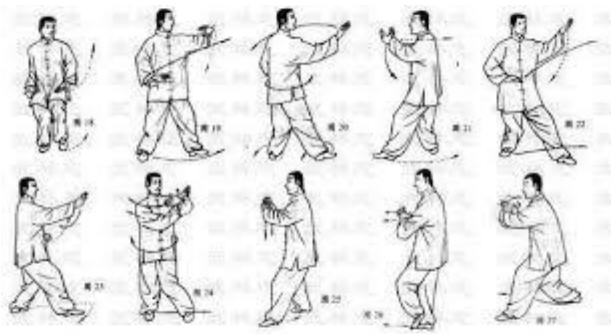
What Are Algorithms?

- Algorithms can be viewed as “程式譜”
    - How to solve computation problems correctly and efficiently
- Similar terms
    - 食譜 (recipes)、樂譜 (sheet music)、劍譜、棋譜、拳譜
    - 臉譜、族譜、光譜、頻譜

Methods

Data, or example based





What Are Data Structures?

- Data structures can be viewed as “everything about data”
  - How to map the real world to the abstract representation?
  - How to use memory effectively?
- Example
  - Six Degrees of Separation (Stanley Milgram, 1960)
  - Facebook users = 1.59 B, DOS=3.57 (link)

- So…
  - 食譜 + 食材 = 菜 (Recipes + Ingredients = Dishes)
  - 樂譜 + 樂器 = 音樂 (Sheet music + Instruments = Music)
  - 劍譜 + 寶劍 = 天下無雙

  ➔ Algorithms + Data Structures = Programs

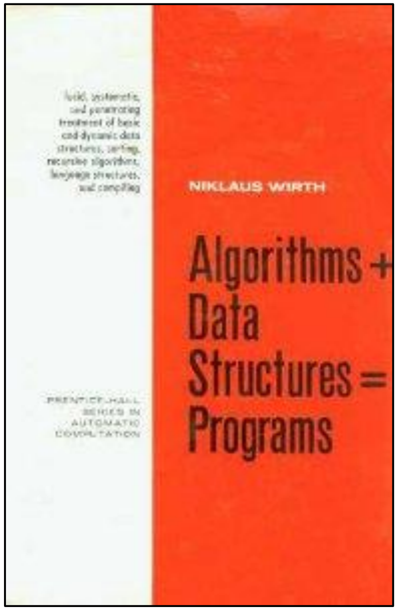
Why Data Structures and Algorithms?

- A good program needs to leverage two types of resources on computers
  - Computing units: CPU, FPU, GPU, etc. ➔ Time
  - Storage units: memory, disks, networks, etc. ➔ Space
- Programs = Algorithms + Data Structures
  - Algorithms focus on computation issues, but needs to be accompanied by proper data structures
  - Data structures focus on storage management, but needs to be accompanied by proper algorithms

DSA helps you write better programs!

Algorithms & Data Structures

- Algorithms + Data Structures = Programs
  - A famous book published in 1976
  - The textbook for my DSA course
- Algorithms (演算法)
  - How to do computation? ➔ Efficient use of CPU
- Data structures (資料結構)
  - How to arrange data? ➔ Effective use of storage
- Trade-offs between computation and storage
  - You'll learn how to trade space with time, or vice versa.


lucid, systematic, and penetrating treatment of basic and dynamic data structures, sorting, recursive algorithms, language structures, and compiling
PRENTICE-HALL SERIES IN AUTOMATIC COMPUTATION
NIKLAUS WIRTH
Algorithms + Data Structures = Programs

About Algorithms

Quiz!

- Five basic criteria of algorithms (by Knuth)
  - Input: Zero or more quantities are externally supplied
  - Output: At least one quantity is produced
  - Definiteness: Each instruction is clear and unambiguous
  - Finiteness: The procedure terminates after a finite number of steps
  - Effectiveness: Each instruction is basic and feasible (do-able by computers)
- How to describe an algorithm
  - English: Description in a natural language
  - Graphic representation: Flow chart
  - Pseudo code: Program-like description in English
  - Programs: C/C++ combined with comments

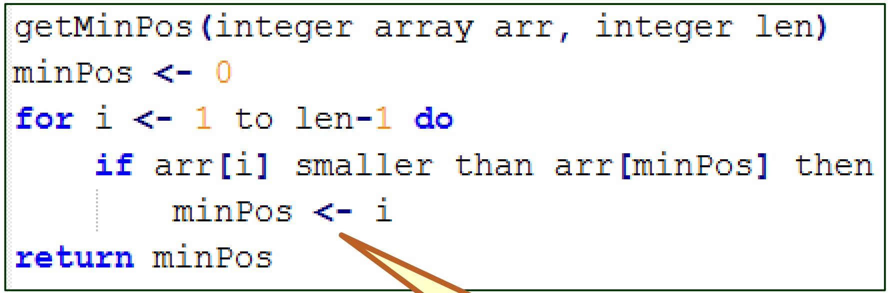
Five Criteria of 食譜

- 食材
  - 番茄、蛋、蔥、薑、太白粉水、鹽、糖
- 食譜：番茄炒蛋
  - 蔥切花、薑切末備用。
  - 番茄去除蒂頭，劃十字刀，下鍋汆燙後去皮。
  - 蛋液打勻，加少許鹽。
  - 番茄切成小塊備用。
  - 太白粉加水備用(1：3.5)。
  - 起油鍋爆香少許薑末，加入番茄、3大匙水、鹽、糖炒勻且湯汁稍微收乾。
  - 加入少許太白粉水勾芡。
  - 再加入蛋液輕輕翻炒。
  - 起鍋前灑上蔥花。

- Criteria
  - Input
    - 食材
  - Output
    - 菜
  - Definiteness
    - 清楚的指令
  - Finiteness
    - 一定可以做完
  - Effectiveness
    - 可行的指令（電腦可完成）

FIVE CRITERIA OF ALGORITHMS

- Pseudo code for finding the index of the smallest number in an array: getMinPos()



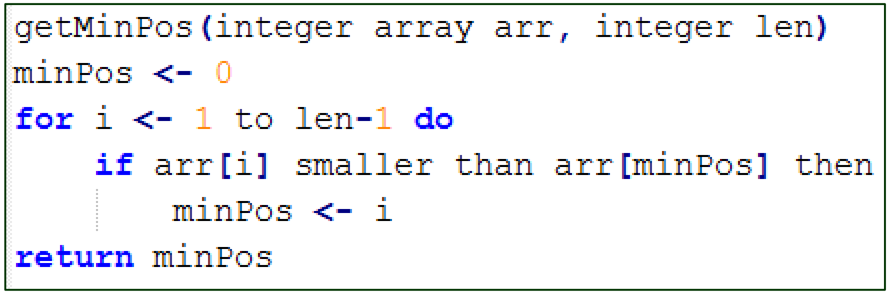
```
getMinPos(integer array arr, integer len)
minPos <- 0
for i <- 1 to len-1 do
    if arr[i] smaller than arr[minPos] then
        minPos <- i
return minPos
```

Assignment

- Criteria
  - Input
    - An array
  - Output
    - Index of the smallest element in an array
  - Definiteness
    - Clear steps
  - Finiteness
    - Will terminate
  - Effectiveness
    - Achievable by computers

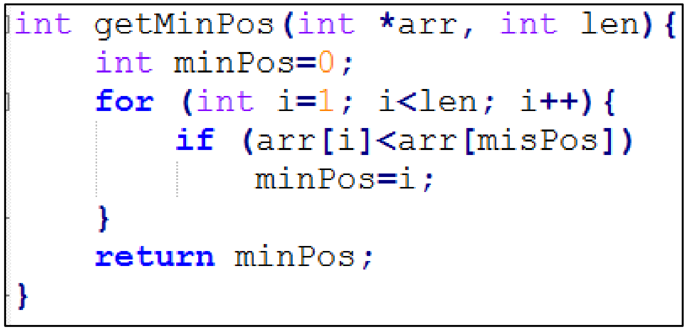
Pseudo Code vs. Real Code

Pseudo code:



```
getMinPos(integer array arr, integer len)
minPos <- 0
for i <- 1 to len-1 do
    if arr[i] smaller than arr[minPos] then
        minPos <- i
return minPos
```

Code in C++:



```
int getMinPos(int *arr, int len){
    int minPos=0;
    for (int i=1; i<len; i++){
        if (arr[i]<arr[misPos])
            minPos=i;
    }
    return minPos;
}
```

- How to prove the correctness of getMinPos()
  - Claim: arr[minPos]<=arr[j] for j=0, 1, …, len-1
  - Claim 2: After iteration i, arr[minPos]<=arr[j] for j=0, 1, …, i
  - Proof by mathematical induction (數學歸納法)
    - Claim 2 holds when i=1
    - Assume claim 2 holds when i=k ➔ Show that claim 2 holds when i=k+1.

Claim 2 is often called Invariance property of loops

Selection Sort

- Animation
  - Flash, YouTube, HTML
  - Wiki about selection sort
- Pseudo code
  - Input: an integer array of length n
  - Output: an in-place sorted array
  - For i from 0 to n-1
    1. Let s_{id} be the index of the smallest number from list[i] to list[n-1]
    2. Interchange list[i] and list[s_{id}]

Important!

- Step 1 can be achieved by getMinPos().
- Step 2 can be done by the computer easily.

Sample Quiz for Selection Sort

- Please show each step of selection sort on the vector:

  3 5 1 4 2 7 9 6 8

Correctness of Selection Sort

- Theorem
  - After the loop of i=q, for any j>q, we have
    - list[0]<list[1]<list[2] … < list[q] < list[j]
- Proof by mathematical induction
  - When q=0, the statement is true
  - Assume statement is true when q=t; then when q = t+1…