Algorithms and Their Applications
- Course Introduction -

Won-Yong Shin

March 16th, 2020

연세대학교
YONSEI UNIVERSITY

- Algorithms & Their Applications: CSE5850

- Instructor: Won-Yong Shin (신원용)
  - Office: ASTC 608A (첨단관608A호)
  - E-mail: wy.shin@yonsei.ac.kr

- Meeting time and location
  - Time: Mon 15:00-17:50pm
  - Location
    - Meeting up via *Zoom* unless otherwise stated
    - (Note: ASTC 516 was originally assigned as a classroom)

- Office hours
  - Available whenever appointments are made via e-mail

- Some requirements
  - **Basic programming experience necessary**
    - Any tool among C, C++, R, Python, Matlab, etc.
  - Some background in Math required
  - Knowledge on data structures would be helpful but not necessarily required
  - You are expected to show up and participate in class

- Lectures
  - Will be delivered in English

Programming Guidelines (*Only for Python Users*) (1/2)

- If you prefer to code on Python,
  - Python 3.7 (for Windows/Linux/Mac)
  - Downloadable via
    - 1) https://www.python.org/downloads/
    - 2) https://www.anaconda.com/distribution/ (*recommended*)
  - What comes up next for the Anaconda installation? ANACONDA.
    - Adaconda Prompt
    - Jupyter Notebook
    - Spyder

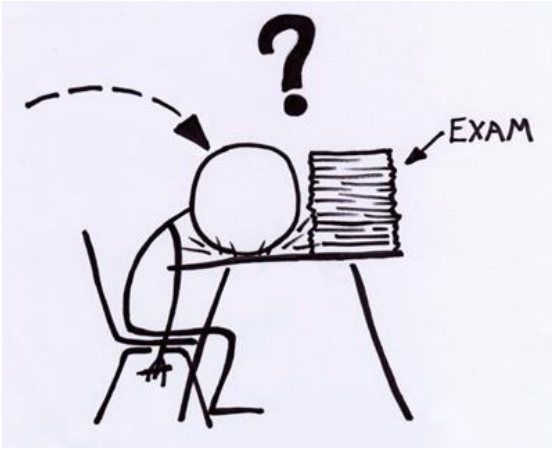
Programming Guidelines (*Only for Python Users*) (2/2)

- To what extent machine learning (ML) packages can be used?
  - 1) **Tensorflow 2.0** is allowed if needed
    - That is, this is optional
    - Can be installed at https://www.tensorflow.org/install
  - 2) Recommended **NOT** to use other frameworks such as Keras, PyTorch, Scikit-learn, etc.
  - 3) Libraries such as Numpy and SciPy can be fully exploited for efficient array computations and numerical analysis

- **Homework**
  - 20% of your grade, assigned approximately once per three weeks
  - You are expected to turn them in a week after they are assigned
    - Being late by one week will have a 20% penalty
    - Being later than one week will result in no credit

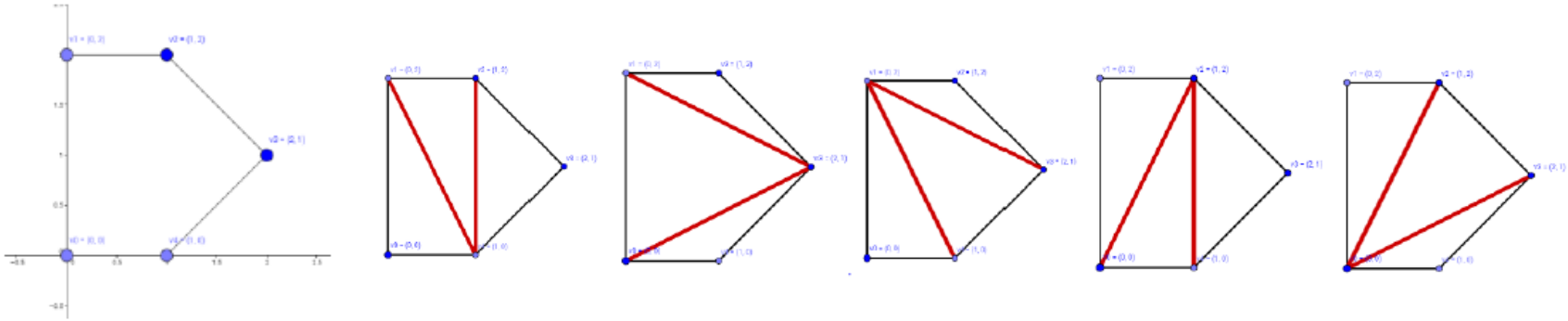
- **Mid-term Exam**
  - 30% of your grade
  - Approx. May 11th (*tentative*)
  - Will be 2 hours

- **Final Project**
  - 50% of your grade



- Make your own algorithm!
  - 1) Define one's own problem by selecting a particular data science or machine learning problem based on (but not limited to) what we have done in class
  - 2) Implement a variant of an algorithm built upon the existing methodology to solve the problem
  - 3) Evaluate and analyze the performance
- Report format will include
  - A pseudo-code representation
  - A source code in any kind
  - Comments
  - Discussions
  - **Note that a proposal should be submitted in advance**
- Presentation
  - **10 mins talk each**
- Evaluation
  - *Numerical* evaluation via a software tool
  - *Analytical* evaluation in terms of time and complexity

- Sample #1
  - Problem: When dividing a polygon into triangles by drawing a new line connecting two vertices that are not adjacent to each other in the shape of the polygon, what is the minimum sum of the lengths of the drawn lines?

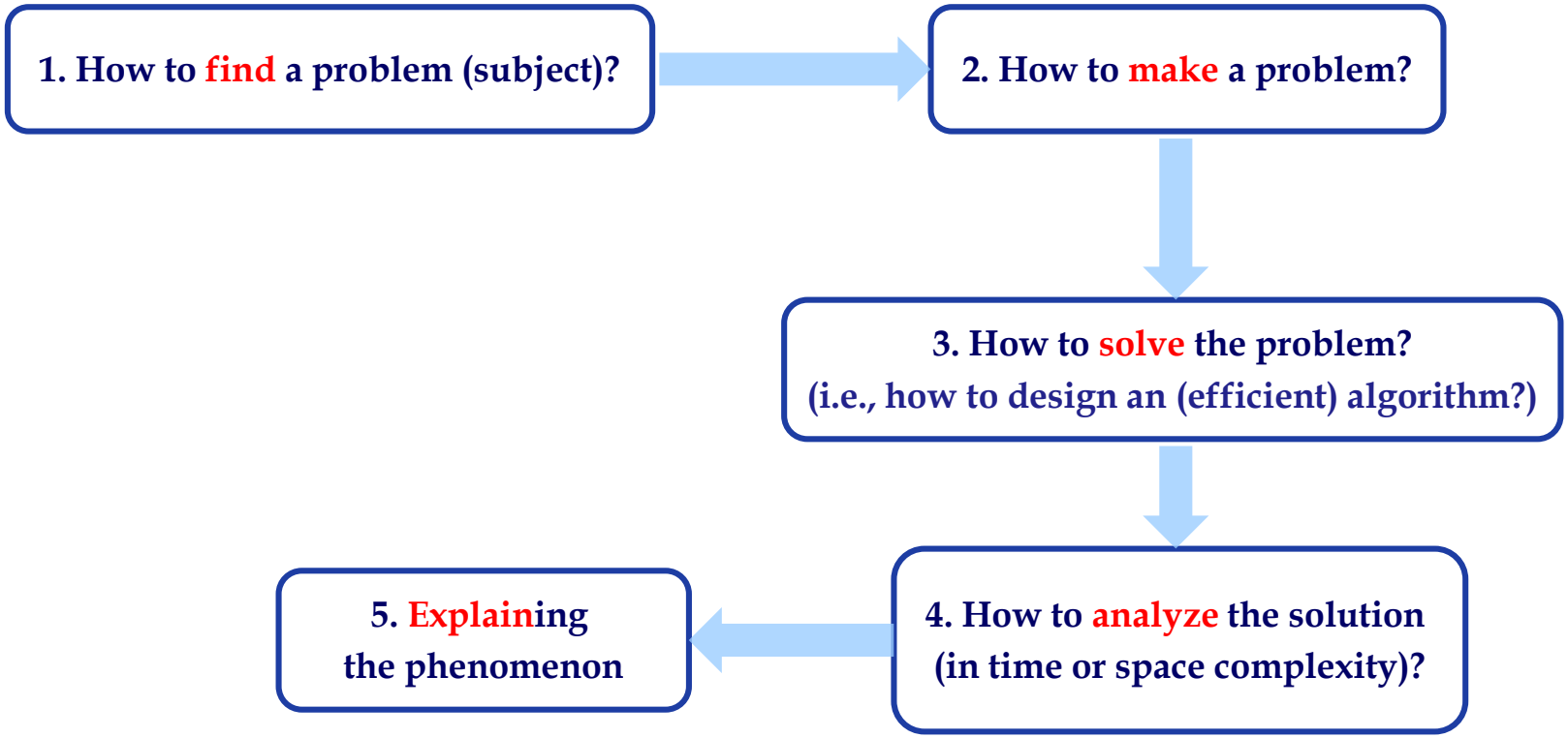


  - Solution? Dynamic programming

- Sample #2
  - Problem: If a positive integer can be expressed as the sum of one ore more prime numbers, what are the total cases?
    - e.g., 41

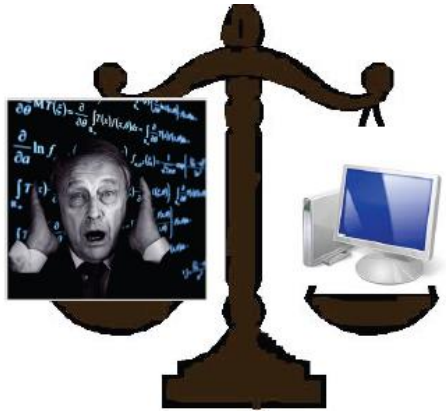
      3 cases: 2+3+5+7+11+13, 11+13+17, 41
  - Solution? Dynamic programming via Sieves of Eratosthenes

- Methodology in Designing algorithms



- Textbook
  - **Introduction to the Design & Analysis of Algorithms** (by Anany Levitin)
- Auxiliary textbook
  - **Introduction to Algorithms** (by Thomas H. Cormen *et al.*)
  - **Data Structures – A Pseudocode Approach with C** (by Richard F. Gilberg and Behrouz A. Forouzan)
- Reference
  - MIT open course:
    - http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-046j-introduction-to-algorithms-sma-5503-fall-2005/video-lectures/

- I will be happy if by the end of the class you …
  - Hold a skill of how to find a proper problem
  - Are capable of designing an effective algorithm to solve the problem
  - Analyze the time and space complexity
  - Have knowledge on the interpretability for real-world applications



- In the syllabus (*tentative*):
  - (1st week) Fundamentals of algorithmic problem solving
  - (2nd week) The analysis framework in algorithm, Introduction to **stacks** in data structures
  - (3rd week) Introduction to **queues** and **linked lists** in data structures
  - (4th week) **Brute force** and exhaustive search
  - (5th week) **Divide-and-conquer**: Introduction and sorting
  - (6th week) **Divide-and-conquer**: Applications
  - (7th week) **Dynamic programming**: Basic examples
  - (8th week) **Dynamic programming**: Applications (e.g., binary search trees)
  - (9th week) **Greedy algorithm**: Elements of the greedy strategy
    - Mid-term exam
  - (10th week) **Greedy algorithm**: Applications and analysis of performance guarantees
    - Project proposal due
  - (11th week) Feedback and comments for the project proposal
  - (12th week) **Heap and heapsort**
  - (13th week) Project: Presentation Part I
  - (14th week) Project: Presentation Part II
  - (15th week) Q&A session (No class)
    - Final report due

Why Should We Take This Class?

- Wikipedia:
  - In math and computer science, an algorithm is an **unambiguous specification** of how to solve a class of problems
  - Algorithms can perform calculation, data processing, and automated reasoning tasks
  - (*Informal definition*) A set of rules that precisely defines a sequence of operations, which would include all computer programs

- Example for finding **gcd**(m, n)

Method #1

- Step 1: Find the prime factorization of m
- Step 2: Find the prime factorization of n
- Step 3: Find all the common prime factors
- Step 4: Compute the product of all the common prime factors and return it as gcd(m,n)

Method #2 (Euclid' algorithm)

- Repeated application of equality gcd(m,n) = gcd(n, m mod n) until the second number becomes 0

```
while n ≠ 0 do
    r ← m mod n
    m ← n
    n ← r
return m
```

Pseudocode

- Understanding the *computation limits*
  - Predict the computational complexity and memory usage of a given program before real implementation

n	$\log_2 n$	n	$n \log_2 n$	n^2	n^3	2^n	$n!$
10	3.3	10^1	$3.3 \cdot 10^1$	10^2	10^3	10^3	$3.6 \cdot 10^6$
10^2	6.6	10^2	$6.6 \cdot 10^2$	10^4	10^6	$1.3 \cdot 10^{30}$	$9.3 \cdot 10^{157}$
10^3	10	10^3	$1.0 \cdot 10^4$	10^6	10^9		
10^4	13	10^4	$1.3 \cdot 10^5$	10^8	10^{12}		
10^5	17	10^5	$1.7 \cdot 10^6$	10^{10}	10^{15}		
10^6	20	10^6	$2.0 \cdot 10^7$	10^{12}	10^{18}		

Table 2.1 Values (some approximate) of several functions important for analysis of algorithms

(by Anany Levitin)

Importance of Algorithms (2/2)

- Becoming crucial when you take job interviews at AI tech companies
  - The most significant subjects:

    Data Structures

    Algorithms

- Hacking a Google interview
  - Reference: http://courses.csail.mit.edu/iap/interview
  - Problem #1: (**Odd Man Out**) You're given an unsorted array of integers where every integer appears exactly twice, except for one integer which appears only once. Write an algorithm (in a language of your choice) that finds the integer that appears only once.
  - Problem #2: (**Path Between Nodes in a Binary Tree**) Design an algorithm to find a path from one node in a binary tree to another.