Lecture #0
Course Overview

Algorithm

JBNU Spring 2021

Jinhong Jung

Outline

Course Overview ☞

Introduction to Algorithm

Course Information

Instructor

- **Name**: Jinhong Jung
- **E-mail**: jinhongjung@jbnu.ac.kr
- **Contact**: ask your questions using **CLASSUM** as possible
  - **CLASSUM** URL: https://www.classum.com/EXSGOJ
  - Let's share our knowledge on this topic
  - Able to anonymize your identification as well!

Lecture Time

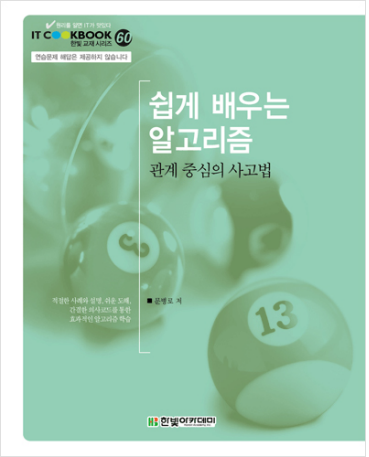
- **(1분반) Class 1**
  - Monday, 11:00-13:00
  - Wednesday, 11:00-12:00
- **(2분반) Class 2**
  - Monday, 14:00-16:00
  - Wednesday, 14:00-15:00

Reference Text

관계 중심의 사고법
쉽게 배우는 알고리즘 (개정판)

- 저자: 문병로
- 출간: 2018
- 출판사: 한빛 아카데미

<2018>



<2013>

Notice

- I'll try to make every lecture slide and homework **self-contained** as possible
  - But some topics/contents aren't going to be included
- Consider buying the book if you want to check details of an algorithm or data structure and solve its related problems

Prerequisites

(★★★) Data structure

- Implement commonly used data structures
  - e.g., list, stack, queue, tree, heap, graph, hash, etc.
- Analyze their time and space complexities at least for worst cases

(★★) Programming skills

- Implement a pseudocode with your own language
  - e.g., C/C++, JAVA, Python, etc.
  - **C++ is recommended** for online judge system
- Analyze, debug, and fix your codes by yourself

(★) Discrete mathematics

- Familiar with mathematical expressions
- Useful when you try to prove a claim logically

Self-diagnosis For This Course

If you are not sure whether "I can follow well this course", answer the following questions:

- **Q1.** What do the below pseudocodes mean?
- **Q2.** Can you translate them in your own programming language (e.g., C++) without any error?

```
def func_A(u):
    visited[u] ← true

    for each v in neighbors of u:
        if visited[v] is false:
            func_A(v)
```

```
def func_B(s):
    visited[s] ← true
    queue.enqueue(s)

    while queue is not empty:
        u ← queue.dequeue()
        for each v in neighbors of u:
            if visited[v] is false:
                visited[v] ← true
                queue.enqueue(v)
```

Cases For Self-diagnosis

Case 1) 😀 I can answer "Yes" for both questions

- ⇒ Do not need to worry about it! Go straight!

Case 2) 😟 I know what they are for, but cannot implement them…

- You will have trouble doing programming assignments
- ⇒ Need to practice programming skills especially for implementing data structures

Case 3) 😰 I cannot answer both questions

- **You are NOT going to follow well every lecture**
  - Especially after the midterm exam
- ⇒ Need to study data structure by yourself (in advance) if you must take this course

Do If You're In Cases 2 & 3

Solve programming problems at an online judge system (https://www.acmicpc.net/step)

- Choose 2~3 representative problems for each step

Topics	Programming	Math	Data Structure
Steps	1, 2, 3, 4, 5, 6, 7, 10	8, 9	17, 18, 21, 23, 27

- **Must finish this self-study by March 30!**
  - Expect you may study during 4~6 hours for each day

If you have a plan to feel angry and dissatisfaction without any effort, do the followings:

- **Step 1)** Open your browser and go to JBNU's Oasis
- **Step 2)** Drop this class without any hesitation

Logistics

Teaching environment

- All lectures are offered online via Zoom, but **exams are taken offline**
  - Each lecture will be recorded, and then uploaded to **YouTube**

Language

- Korean is used by default in all aspects **except for lecture slides** which will be written in **English**

Notification and deadline policy

- All notifications, schedules, and materials are uploaded to
  - LMS: https://ieilms.jbnu.ac.kr
- ⚠️ **No plan to accept any submission after its deadline**
  - Please check the bulletin board regularly and do your homework in advance!

Evaluation Plan (1)

Grading policy: relative evaluation (상대평가)

- The ratio for Grades A and B should be 80%

Grade A	Grade B	Grade C	Others
40±5%	40±5%	15+5%	5%

Proportion for grading

Attendance	Homework	Midterm Exam	Final Exam
5%	25%	35%	35%

- The maximum total score is 100 points
- I'll manage two classes for this course, but evaluate them as if all of students are in one class

Evaluation Plan (2)

Disqualification policies

- Any form of **cheating** (e.g., homework and exam) will give the corresponding task of its contributors **a zero point**
  - ⚠️ It could be reported to **our school's disciplinary committee**

- If you have absences of **more than a quarter (1/4) of all lecture time**, then **Grade F** will be assigned immediately
  - This course will take 45 hours in this semester
  - Thus, the absence of 11.25 hours presents Grade F to you
  - ⚠️ Monday class takes 2 hours, and Wednesday one takes 1 hour

- If your total score is **less than 10 points**, then **Grade F** will be assigned immediately

Schedule (Tentative)

Week	Content	Week	Content
1	Introduction	9	Graph Algorithm I
2	⁕ Recurrence & Math	10	† Graph Algorithm II
3	Sort I	11	Greedy Algorithm
4	† Sort II	12	† String Matching
5	Selection & ADB I	13	⁕ NP
6	† ADB II	14	State Space Tree
7	† Dynamic Programming	15	Final Exame
8	Midterm Exam		

- ADB stands for advanced data structures *(such as balanced BST, disjoint set, etc.)*
- † indicates it has a programming assignment *(tentative)*
- ⁕ indicates it has a report assignment *(tentative)*

- ⚠️ **No plan to review programming skills and details of basic data structures in this course!**

Outline

Course Overview

Introduction to Algorithm ☞

What Is Algorithm?

To describe the sequential process for solving a problem using a computer(s)

- From the input data to the final output

Input

스펀지케이크(20×20cm) 1개, 크림치즈 200g, 달걀 푼 물 2개 분량, 설탕 3큰술, 레몬즙·바닐라에센스 1큰술씩, 딸기시럽(딸기 500g, 설탕 1½ 컵, 레몬즙 1작은술), 딸기 1개, 플레인 요구르트 2큰술

First, clarify the input and output for your target problem

Algorithm

① 케이크 틀의 가장자리에 필름을 돌린 다음 스펀지케이크를 놓는다.

② 볼에 크림치즈를 넣고 거품기로 젓다가 달걀 푼 물과 설탕 3큰술을 세번에 나누어 넣으면서 크림 상태로 만든다.

③ ②에 레몬즙과 바닐라에센스를 넣고 살짝 저은 다음 ①에 붓는다. 이것을 180℃의 오븐에 넣고 20분 정도 굽는다.

④ 냄비에 슬라이스한 딸기와 설탕 1½ 컵을 넣고 끓이다가 약한 불에서 눌어붙지 않도록 저으면서 거품을 걷어낸다. 되직해지면 레몬즙을 넣고 차게 식힌다.

⑤ 접시에 치즈케이크를 한 조각 담고 ④의 시럽을 뿌린 다음 플레인 요구르트와 딸기를 얹어낸다 .

Output

Very Simple Example

Problem: Find a maximum of 100 students' scores

- **Step 1) clarify the input and output**
  - **What is the input?** ⇒ (Informal) 100 scores
    - (Formal) An array $\mathbf{x}$ of size 100, containing the input scores (i.e., $\mathbf{x}[1], \cdots, \mathbf{x}[100]$)
  - **What is the output?** ⇒ (Informal) A maximum of the input scores
    - (Formal) A maximum value among $\mathbf{x}[1], \cdots, \mathbf{x}[100]$
- **Step 2) describe an algorithm for solving the problem**
  - Given the input, it should correctly produce the output

Algorithm written in pseudo-code

```
def max_score(x):
    for each value in x:
        mark the value as maximum if it > the previously marked one
    return the last marked value
```

Algorithm Should Be…

Efficient and correctly designed

- Your algorithm should be practically **efficient for large input** when a modern **computer** is used
- Your algorithm must produce **correct answers** (or output)

Easy-to-understand

- Easily able to implement your algorithm described in a pseudocode using a specific programming language
- Need to consider the abstraction level of target readers

Abstraction level up

Abstraction level down

```
find a maximum of x by looking every value in the array
```

```
for each value in x:
    mark the value as maximum if it > the previously marked one
```

```
for i in range(1, len(x)+1):
    if x[i] > maximum:
        maximum = x[i]
```

Purpose Of This Course

Goal 1) Correctness and efficiency

- To understand how to design and analyze an algorithm in terms of correctness and efficiency

Goal 2) Training using classical problems

- To learn ideas that effectively resolve challenges behind various classical problems

Goal 3) Computational thinking

- To improve your computational thinking so that you can solve new problems by yourself

“How To Think” Matters

In this course, you will study

- Algorithms of various classical problems
  - Developed by some people born in the generation of your grandparents
- Of course, it’s also important to study their solutions
  - Especially for your exams and future coding test

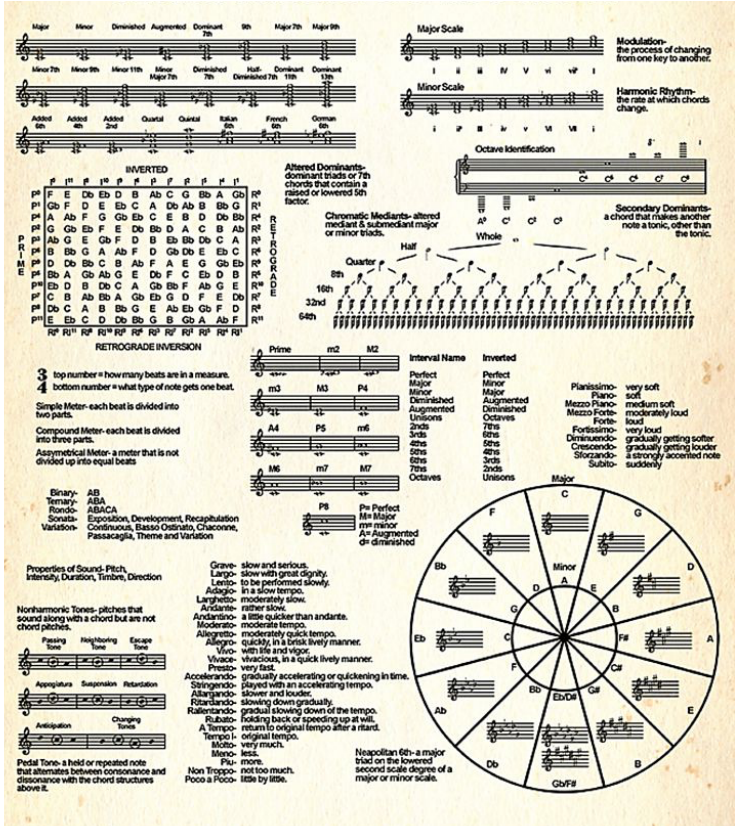
The most important thing is

- To study **how they think** to solve the problems
- Sharply **observe their ideas or principles** behind the solutions and **check why they are working**
- Will be useful building blocks which can be used for **solving new problems**

Do Not Have False Expectation

It covers the theoretical area of computer science

- Thus, it's hard to guarantee that you can pass a coding test even though you get Grade A in this course



You'll learn about such things…

If you want to do coding well,

Try to solve programming problems as many as possible BY YOURSELF!

In Next Lecture

Motivation to algorithm analysis

- Why should we analyze algorithms?
  - Especially, in terms of efficiency

How to analyze algorithms?

- Concept of computational complexity
- Asymptotic analysis (e.g., Big-O notation)