

# Introduction to Algorithms

**Course information**

**Instructor : Kilho Lee**

# Who Am I?

- Kilho Lee (이길호)
  - Venture hall #709
  - Homepage: <https://sites.google.com/view/khlee>
  - Email: [khlee.cs@ssu.ac.kr](mailto:khlee.cs@ssu.ac.kr)
  - Office hours: by appointment
- **MISys lab**: Mobility Intelligence and Computing Systems Lab
  - Research interests
    - Autonomous vehicles
    - Cyber-physical systems/ Internet of Things
    - (Embedded) Real-time systems

# Outline

- Course Info
- Course overview

# Course Information

- 알고리즘 (가)/(나), Introduction to Algorithms
  - 분반 통합 학점 부여
- Lecture time & place
  - Online, pre-recorded video clips.
  - 2~3 videos a week up to 150 min.
    - The LMS system ([class.ssu.ac.kr](http://class.ssu.ac.kr))
- Prerequisites
  - Data Structures is strongly recommended
  - Programming languages:
    - **C++** for examples and programming assignments
    - Python for some examples instead of pseudo codes.

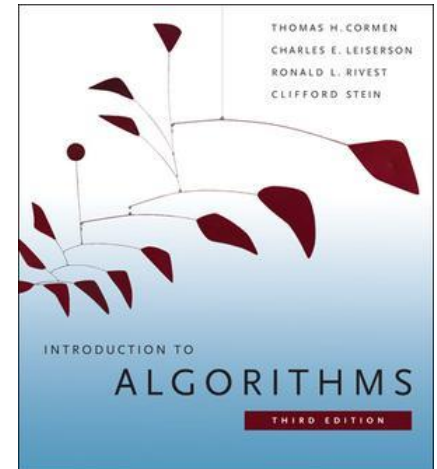
# Course Information

- Textbook

- Introduction to Algorithms (**CLRS**), Third Edition

- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein
    - The latest edition is preferred, but any edition is fine
    - Available for free ebook via the SSU library
    - Translated version also available

- 쉽게 배우는 알고리즘 (문병로)



- Acknowledgement

- The lecture materials are built upon previous efforts of

- Instructor David Eng (Stanford University)
    - Instructor Mary Wootters (Stanford University)
    - Professor H. Chwa (DGIST)

# Course Information

- Grading
  - Homework: 20%
  - Midterm: 35%
  - Final: 35%
  - Class participation: 10%

# Course Information

- Homework (20%)
  - 4~6 quizzes & programming assignments
    - PS (problem solving) style
  - Late submission penalties
    - Allowed to submit **up to 48 hours** after deadline.
    - Up to +24H (1d) : **-25%** penalty
    - Up to +48H (2d) : **-50%** penalty
    - Later than +48H: **-100%** penalty
  - We will be serious on any kind of plagiarism.
- Mid-term exam (35%), Final exam (35%)
  - Planned as **offline**, 대면 시험
  - Any kind of cheating will bring a serious penalty

# Course Information

- Participation (10%)
  - Finish the video clip **at least 90%** within the period.
  - 각 콘텐츠의 학습 기간 이내에 청취 **(90% 이상)** 완료.
  - 결석 1회당 -1P
    - 지각 불인정. 실수/착오 (89.99% 청취, 0.1초 지각 등) 불인정.
- Note
  - 교육부 지침에 따라 1/3 이상 결석 시 학점부여 불가 (F학점)
  - 유고 결석 시 SAINT 시스템을 통해 신청

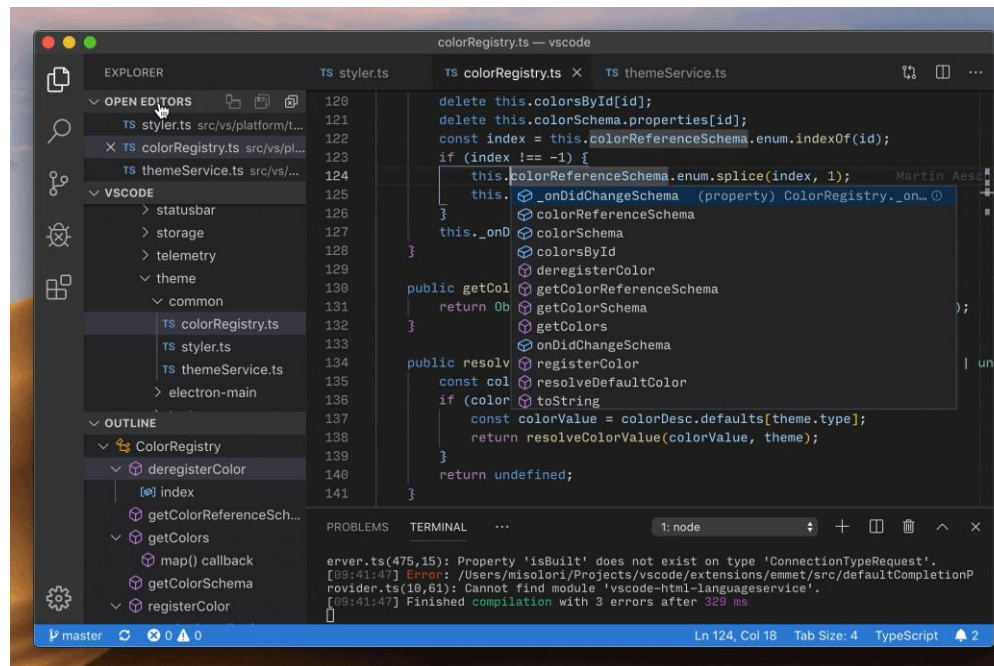


# Course Information

- Communication
  - By bulletin board (Q&A)
  - By email

# Development environment setup

- VSCode + MinGW (g++)
  - 고급프로그래밍및실습 과목과 동일한 설정
  - Windows: LMS 영상 참고
  - MAC/Linux: VSCode/g++ 설정해서 사용 (LMS 영상 및 검색 활용) 하거나, 다른 개발환경 (vim/g++, sublime/g++ 등)



# Introduction to Algorithms

## **L0. Overview**

**Instructor : Kilho Lee**

# **Introduction to Algorithms**

# Computer Science

Computer Science is \_\_\_\_\_

# Computer Science

**Computer Science is abstraction**

The single most important concept in computer science is abstraction

# Abstraction

Abstraction is

---

# Abstraction

**Abstraction is to create a new model  
that allows to ignore irrelevant details**



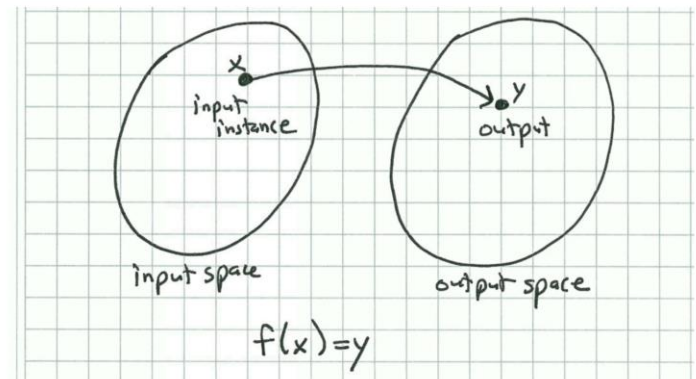
# What is an Algorithm?

- Mathematical **abstraction** of a computer program
  - At the heart of programs lie algorithms
- Well-specified procedure for solving a computational problem
- Computational problem = mapping from inputs to outputs
  - Ex)
    - Given **array of integers**, produce a **sorted array**
    - Given a **graph and nodes s and t**, find a **shortest path from s to t**
    - Given an **integer**, find its **prime factors**

Input space = set of possible inputs

Input instance = a particular input  
(aka a problem instance)

Output space = set of possible outputs

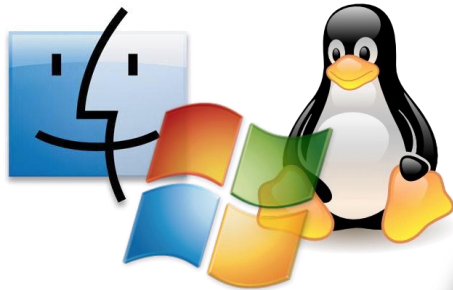


# Why are you here?

- Algorithms are fundamental.
- Algorithms are useful.
- Algorithms are fun.
- It is a required course.

# Why are you here?

- Algorithms are fundamental



Operating Systems



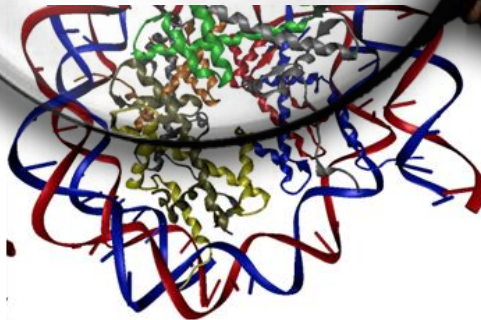
## The Computational Lens



Cryptography



Networking



Computational Biology

Mapping/Navigation  
Image Search  
Web processing  
Streaming video  
Games (graphics rendering...)  
Big data (querying, learning...)

# Why are you here?

- Algorithms are useful

- As we get more and more data and problem sizes get bigger and bigger, algorithms become more and more important.
- Will help you get a job.



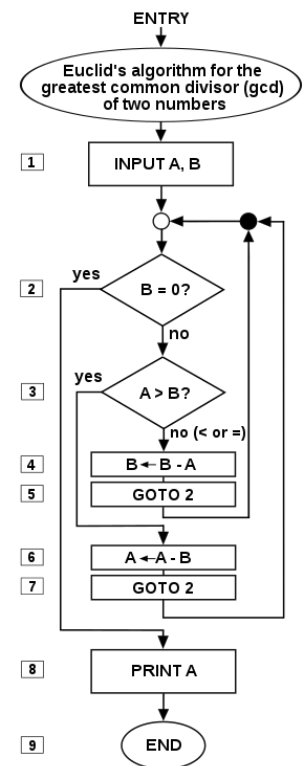
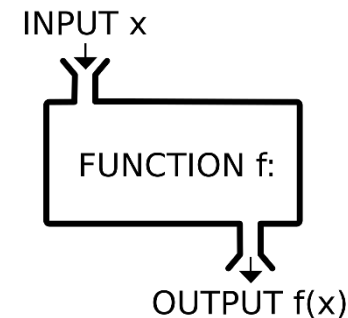
- Algorithms are fun

- Algorithm design is both an **art** and a **science**.
  - Requires a combination of **creativity** and mathematical **precision**



# Course Goals

- Formal definition of algorithm
  - An **algorithm** is a sequence of computational procedures that *transform the input into the output*
- The **design** and **analysis** of algorithms
  - These go hand-in-hand
- Implementation of algorithms



# Course Goals

- **Analysis of Algorithm**

- Does an algorithm actually work?

- **Correctness**

- Is it fast? (Does there exist any better algorithm?)

- **Time/Space efficiency**

- Lower bounds

- Optimality

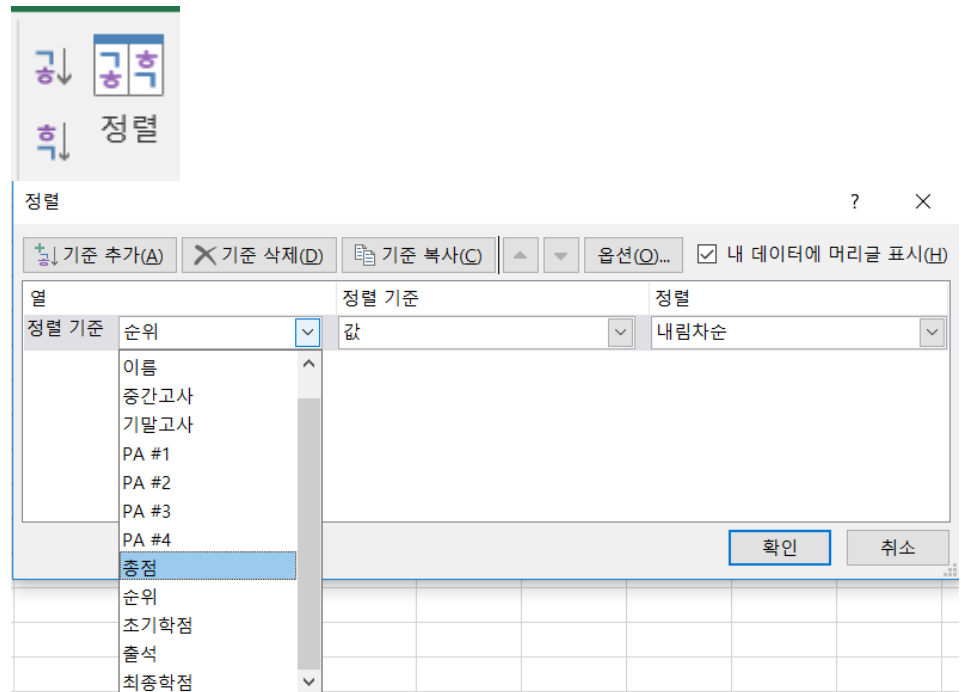
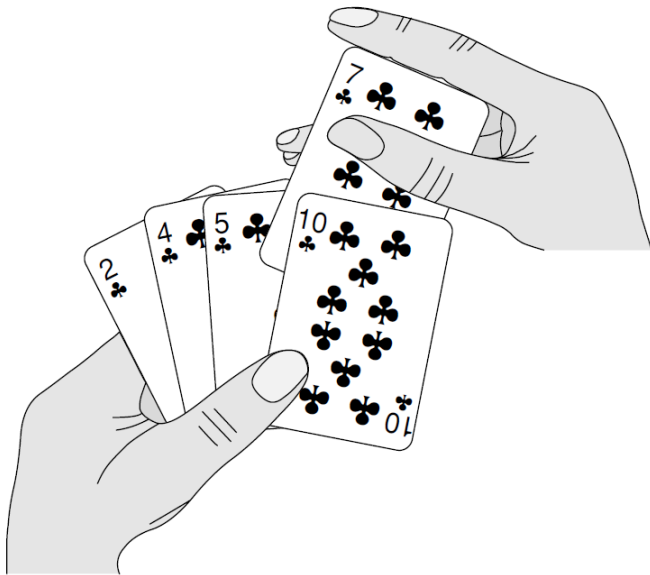
# Course Topics

- Algorithmic Analysis
- Divide and Conquer
- Randomized Algorithms
- Tree Algorithms
- Graph Algorithms
- Dynamic Programming
- Greedy Algorithms
- Advanced Algorithms
- NP-completeness

# Sorting Problem

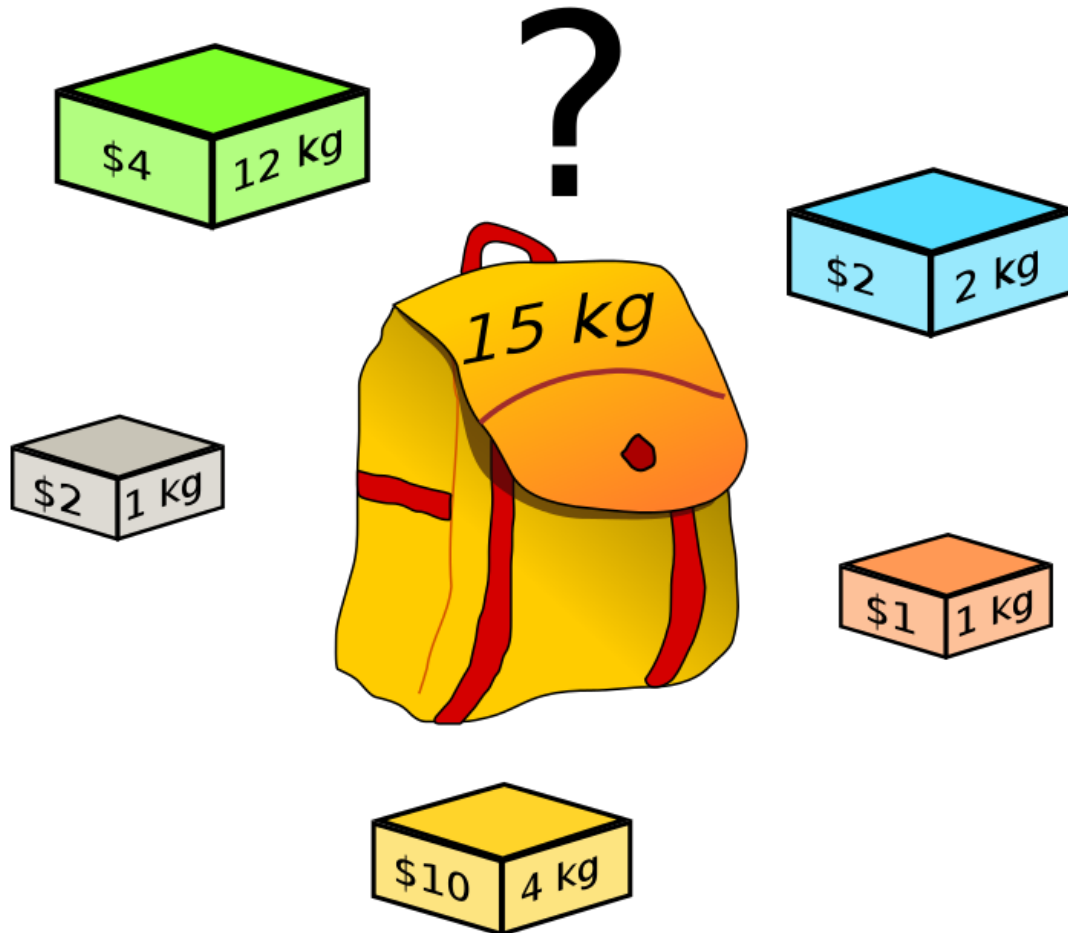
- Sorting problem

- Input: A sequence of  $n$  numbers  $\langle a_1, a_2, \dots, a_n \rangle$
- Output: A permutation (reordering) of the input sequence,  $\langle b_1, b_2, \dots, b_n \rangle$ , such that  $b_1 \leq b_2 \leq \dots \leq b_n$
- Instance:  $\langle 7, 10, 4, 5, 2 \rangle$





# Knapsack Problem

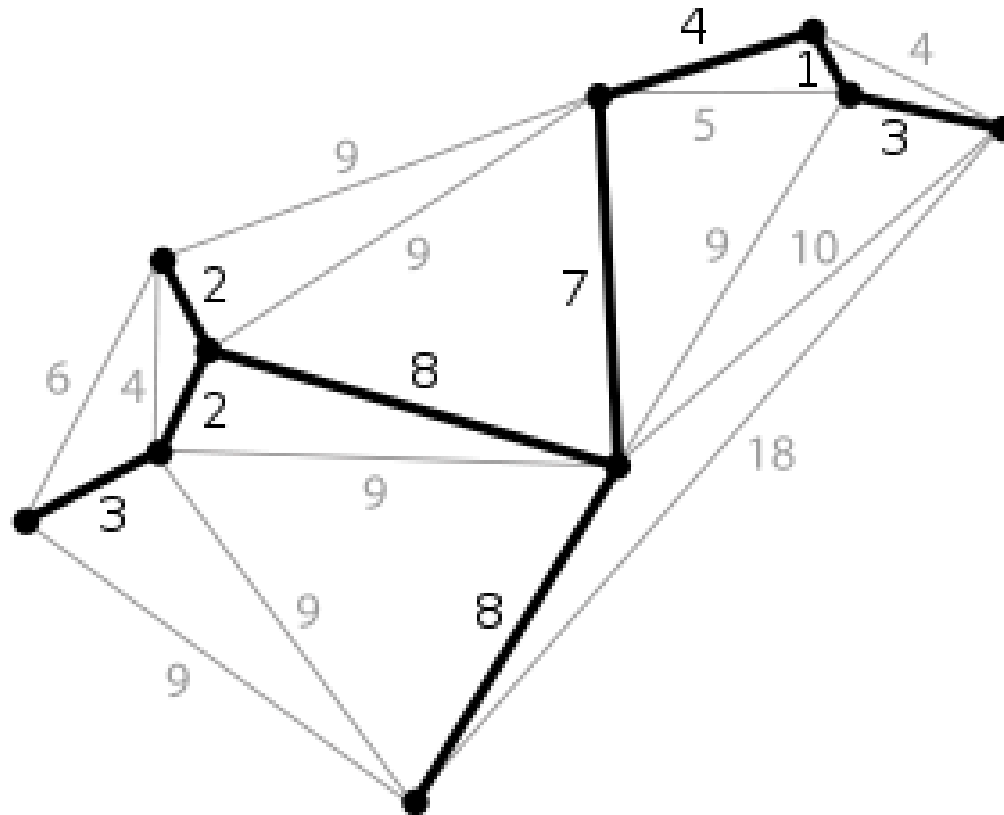


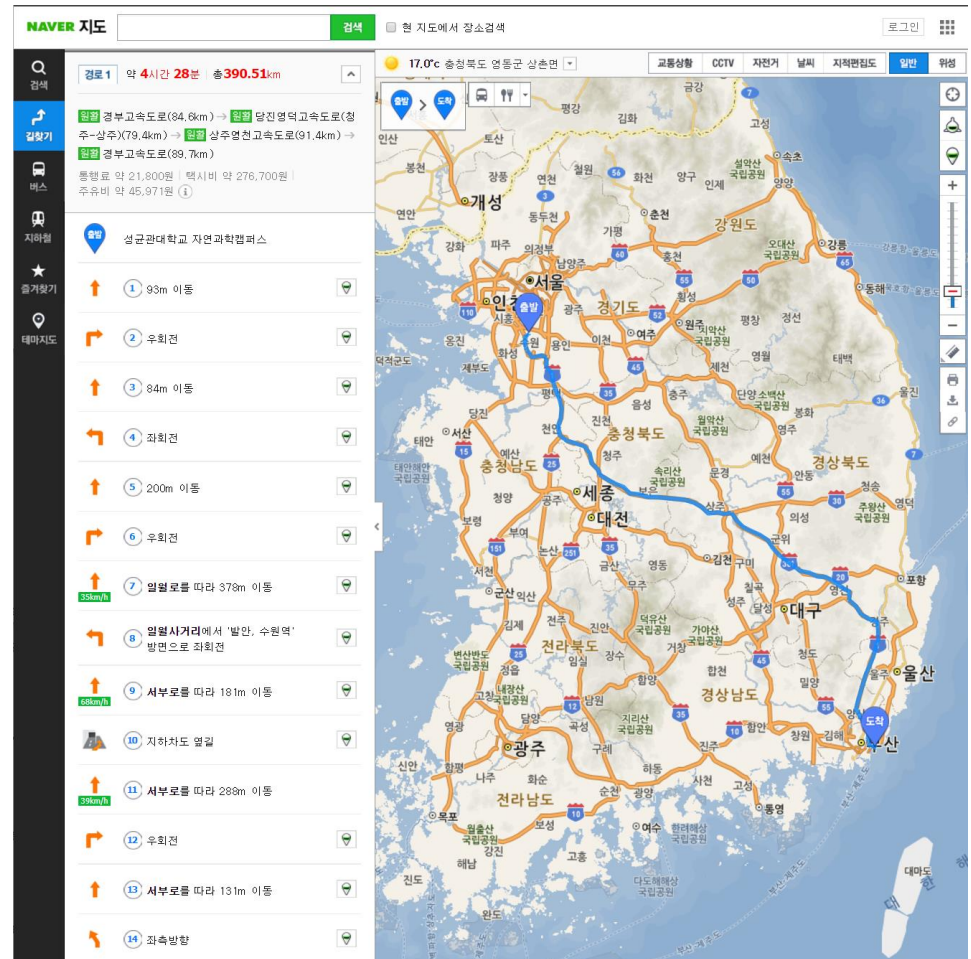
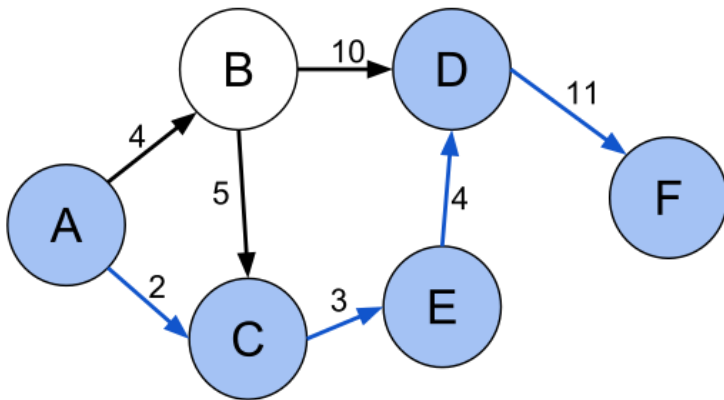
# Edit Distance

		m	o	n	k	e	y
	0	1	2	3	4	5	6
m	1	0	1	2	3	4	5
o	2	1	0	1	2	3	4
n	3	2	1	0	1	2	3
e	4	3	2	1	1	1	2
y	5	4	3	2	2	2	1

Figure from  
<https://vinayakgarg.wordpress.com/2012/12/10/edit-distance-using-dynamic-programming/>

# Minimum Spanning Trees (MST)



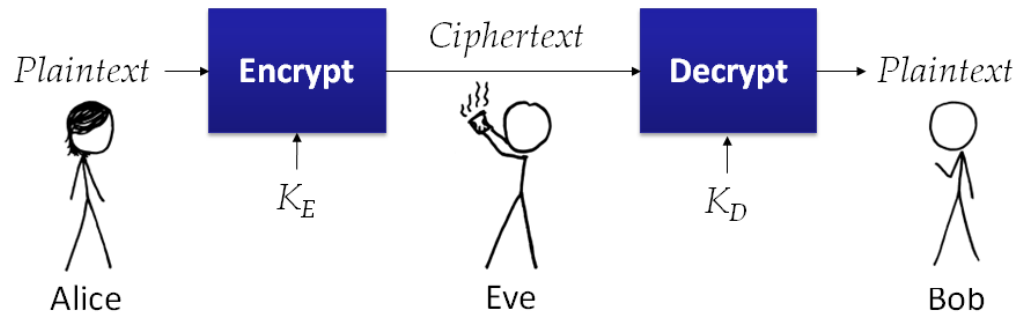
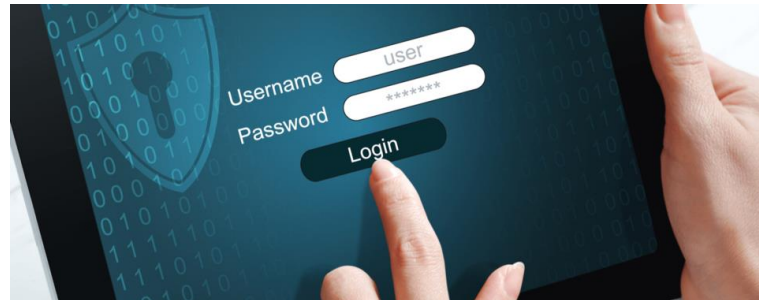


Figures from Wikipedia & NAVER

# Security

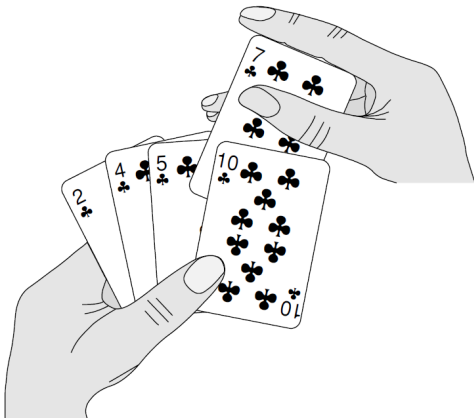
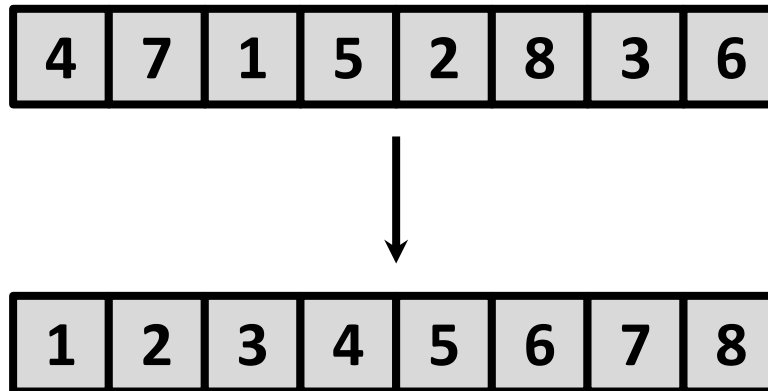
- Cryptography
- Authentication

```
.....  
e is 65537 (0x010001)  
-----BEGIN RSA PRIVATE KEY-----  
MIIEJwIBAAKCAgEApH6mJKb9/XEonQI04LZK2nWydsyZZgzDGTctJLEdm5vA8KB  
D6gCoWJhRURF7fRYDjok4yyf3RCLx8TtpSVLQqGf/QpkQCU5Vmo60ePh33aCusQ  
kVieL9u/fGVE3TAVgicvHu8a071ABHBpc0RDJYr8H3aNkDNN0AojzRs33dfN3nT9  
Pk19e4MlmsJyZhaSRmIGwpQ5hxPWV5081WZdj3hjkQqsq+fhIjcpB6ZLztF8XlyI  
g5Vru3+mx6QFKrltejia09S0kPnZk35dErzcKcJ92xpqMNB66L86qtXCGZ4/30ie  
zGrHCq+fb0F4FyenH14TpbPTZL2N53eYxhfB+WHRDxszmwrQbm3og4LTmwHuouxh  
CNmG4UmIpHgpdWS5q/nKJLmbil7IWvGCKAVNoCyNGZwDLJ9Qr-TXRSb0YUTgnpNma  
3VXPLIu1NePjTzjaW9j+RIilpySkll94Z0MwYyMSHhWbEaDEQBELzQZF+5XS+Swj  
hRLe83ImPz0xUQN5GvEcycovJPkR8XpngHg20HXdM61pISip6vk8wZA3jFZdLvSb  
MWJK6sEWL4wGfqNjNpdHtnMPo0/u6eioslRxqNiGBfqxBtzWXUKdbau8fM3GH9Tw  
b00d9afHejGC8t68T8AcsT0eMduNB0LuG2CqDF/5HT2pZ3xeQcLI99h0d0CAwEA  
AQKCAg8d7rSRWYrQnz3B6tr0FyThe05d6JfwXnLKf0eafEmbBBA26E3f4mA/tz  
SLYCG/XsqedGksT3R+uK0Do8g/9mILqL1rGly98620hM8qBTjtpkQbdESIAmbb/7  
GRkp9corWJTf5UnlszxLTKXq3KA7YB3JQnRvnnp9mBbT5+nXwSAC+3pFfcCAPsr  
VL5e7aS57GBP3LS9HebY0UEu1/R/b1dperUf6VgFH+pjpzbjWSnvc921GygtVRX
```



# An example: Sorting

- Sorting algorithms order sequences of values.
  - For the sake of clarity, we'll pretend all elements are distinct.



# A sorting algorithm

```
def sort(A):  
    for i in range(1, len(A)):  
        cur_value = A[i]  
        j = i - 1  
        while j >= 0 and A[j] > cur_value:  
            A[j+1] = A[j]  
            j -= 1  
        A[j+1] = cur_value
```

# Summary

- Course Goals
  - What is an algorithm?
  - Why we study algorithms?
    - Fundamental, useful, fun
  - Goal: the **design** and **analysis** of algorithms