Analysis and Design of Algorithms

Introduction

Instructor: Morteza Zakeri

Lecturer

- Morteza Zakeri
  - Ph.D. in Computer Engineering (Software),
  - Iran University of Science and Technology (IUST)
  - https://m-zakeri.github.io
- Member of
  - Software Reverse Engineering Research Laboratory
    - http://reverse.iust.ac.ir
  - Association for Computing Machinery (ACM)
- Interested in
  - Program analysis and transformation
  - Software testing and quality assurance
  - Artificial intelligence for software engineering (AI4SE)

About the IUST

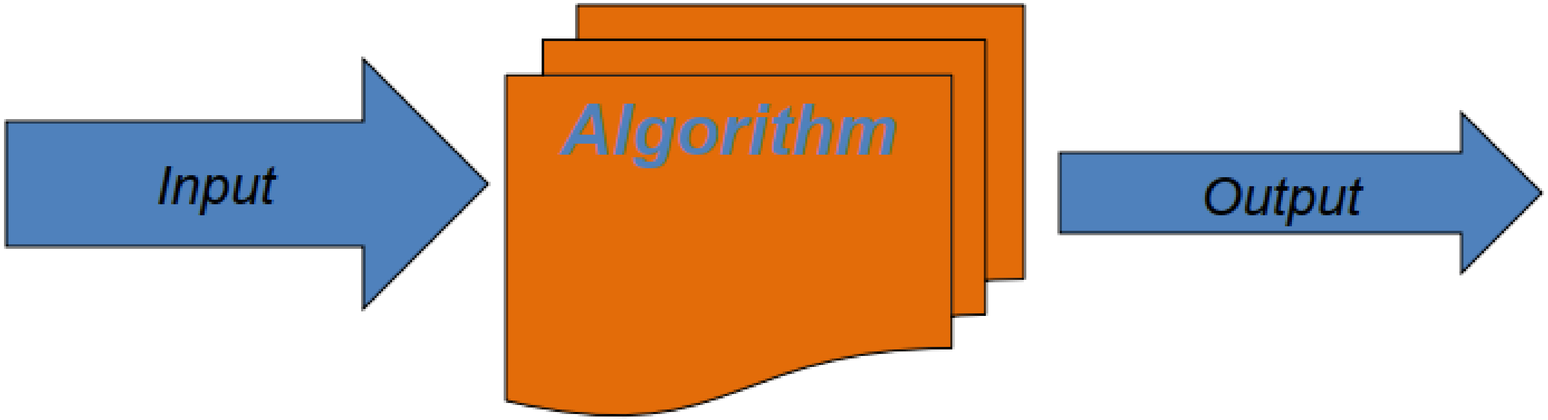
- The university is among the top 4 university in Iran
- Located at Tehran city (Narmak)

What is an algorithm?

- An *algorithm* is a finite set of precise instructions (well-defined computational procedure) for performing a computation or for solving a problem.
  - Takes some value, or set of values, as **input.**
  - Produces some value, or set of values, as **output.**

- A sequence of computational steps that transform the input into the output.

What is an algorithm?



Typically, an algorithm must also ***halt*****.**

Type of typical algorithm inputs

- Literals (Strings, numbers, and so on)
- An array/ vector
- A matrix
- A graph
- ???

Goals of this Course

- **Algorithms**
  - **Design -** How do you *create* an algorithm?
  - **Analysis** – How *efficient* is it?
  - **Correctness** – How sure are you that it works for all input?

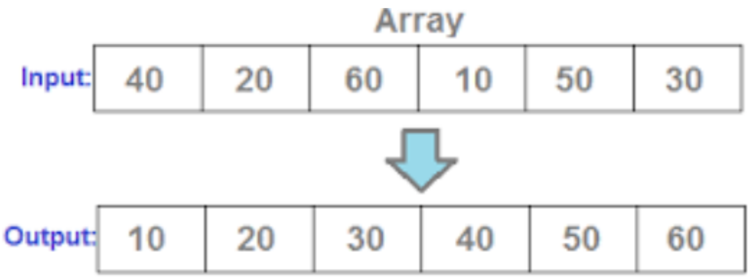
- **Data Structures**
  - Role in efficient algorithms
  - Data structures for common problems
  - **Computational problems**

Computational Problem

- We will look at several recurring problems in a vast set of application domains:
  - Networking, AI, data mining, graphics, manufacturing, etc.
- Sample problems:
  - Sort a sequence of numbers.
  - Given a sorted set of numbers, determine whether a given number exists in the set.
  - Given a set of points in the plane, find the closest two.
  - Given a set of locations, find a route that visits each Location once.
  - **etc.**

Some Well-known Computational Problems

- Sorting
- Searching
- Shortest paths in a graph
- Traveling salesman problem
- Knapsack problem



Array

Input:	40	20	60	10	50	30
Output:	10	20	30	40	50	60

Search

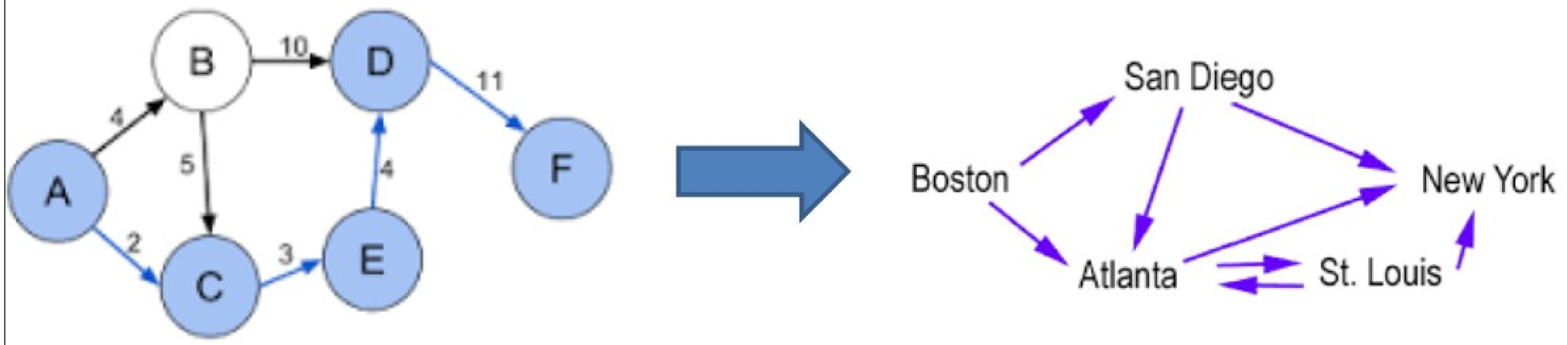
- Search

Text Data

Graph Data

Shortest paths in a graph

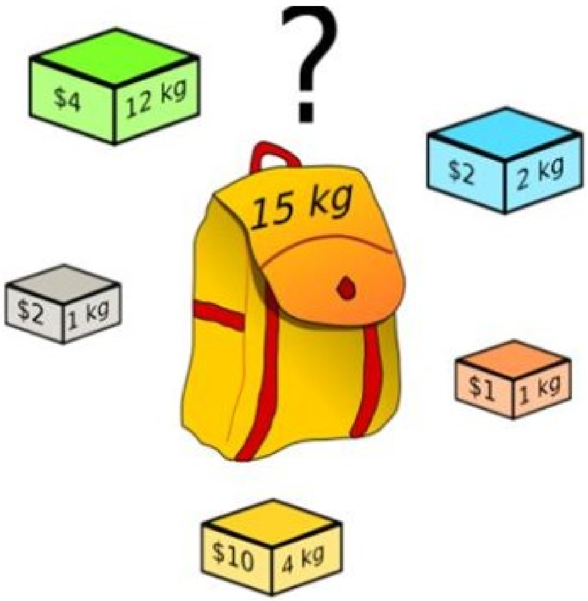
- Snapp, Travel salesperson



Travel Salesman Problem

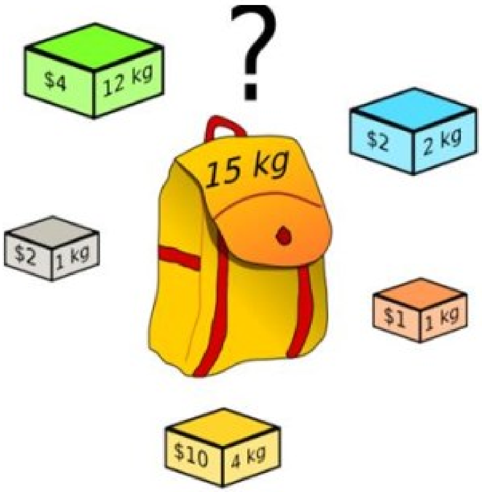
Knapsack problem

- You have a knapsack that has the capacity (weight) W
- You have several Items
- Each item has weight w_i and weight b_j
- You want to put items in the knapsack so that
  - Knapsack capacity is not exceeded
  - Total benefits maximizes

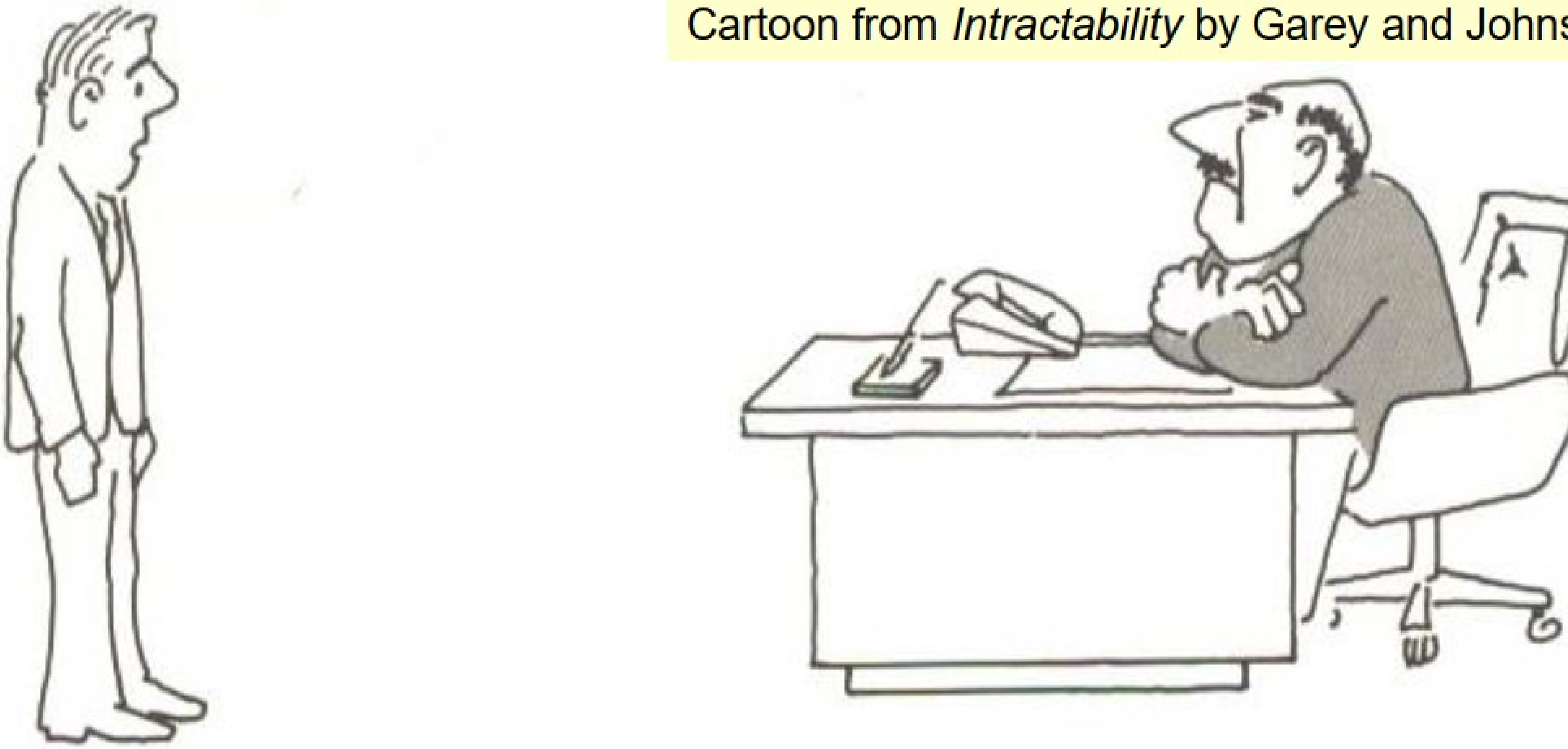


Knapsack problem Types

- 0-1 knapsack problem
  - Items can not be divided
- Fractional knapsack problem
- – For instance, items are liquid or powder



Why should I care about Algorithms?



Cartoon from *Intractability* by Garey and Johnson

"I can't find an efficient algorithm, I guess I'm just too dumb."

Basic issues related to algorithms

- How to design algorithms
  - How to express algorithms
  - Proving correctness
- Efficiency
  - Theoretical analysis
  - Empirical analysis
- Optimality

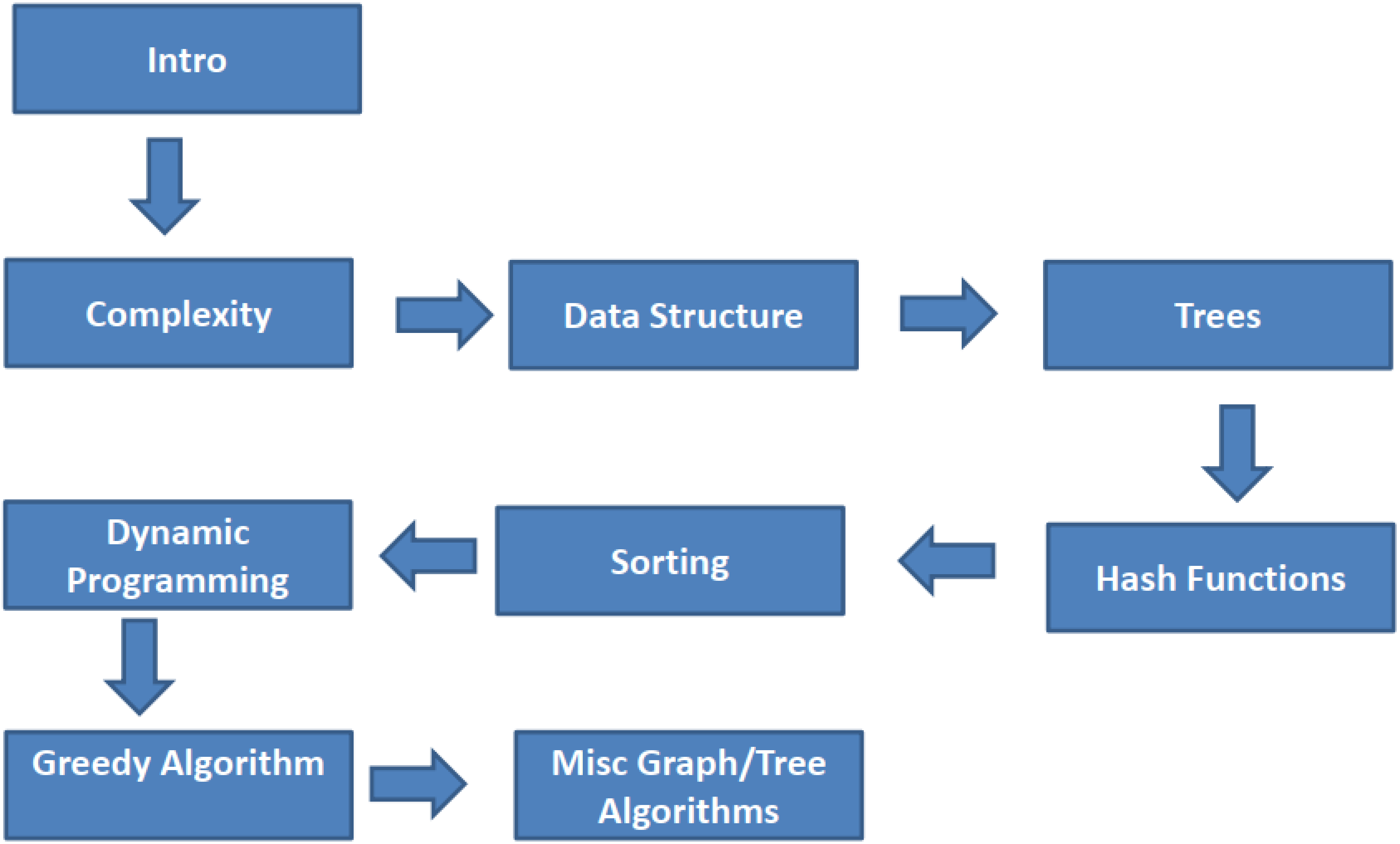
Basic issues related to algorithms

- **Amount of space used:** The amount of space used can be measured similarly. Consideration of this efficiency is often important.
- **Simplicity, clarity:** Sometimes, complicated and long algorithms can be simplified and shortened by the use of recursive calls.
- **Optimality:** For some algorithms, you can argue that they are the best in terms of either amount of time used or amount of space used. There are also problems for which you cannot hope to have efficient algorithms.

Algorithm design strategies

1. Brute force
    1. Proof by exhaustion, proof by cases, proof of each of the cases.
2. Divide and conquer
3. Greedy approach
4. Dynamic programming
5. Backtracking
6. Branch and bound
7. Space and time tradeoffs
8. Heuristics-based algorithms

Our course (big picture)



Intro

Complexity

Data Structure

Trees

Hash Functions

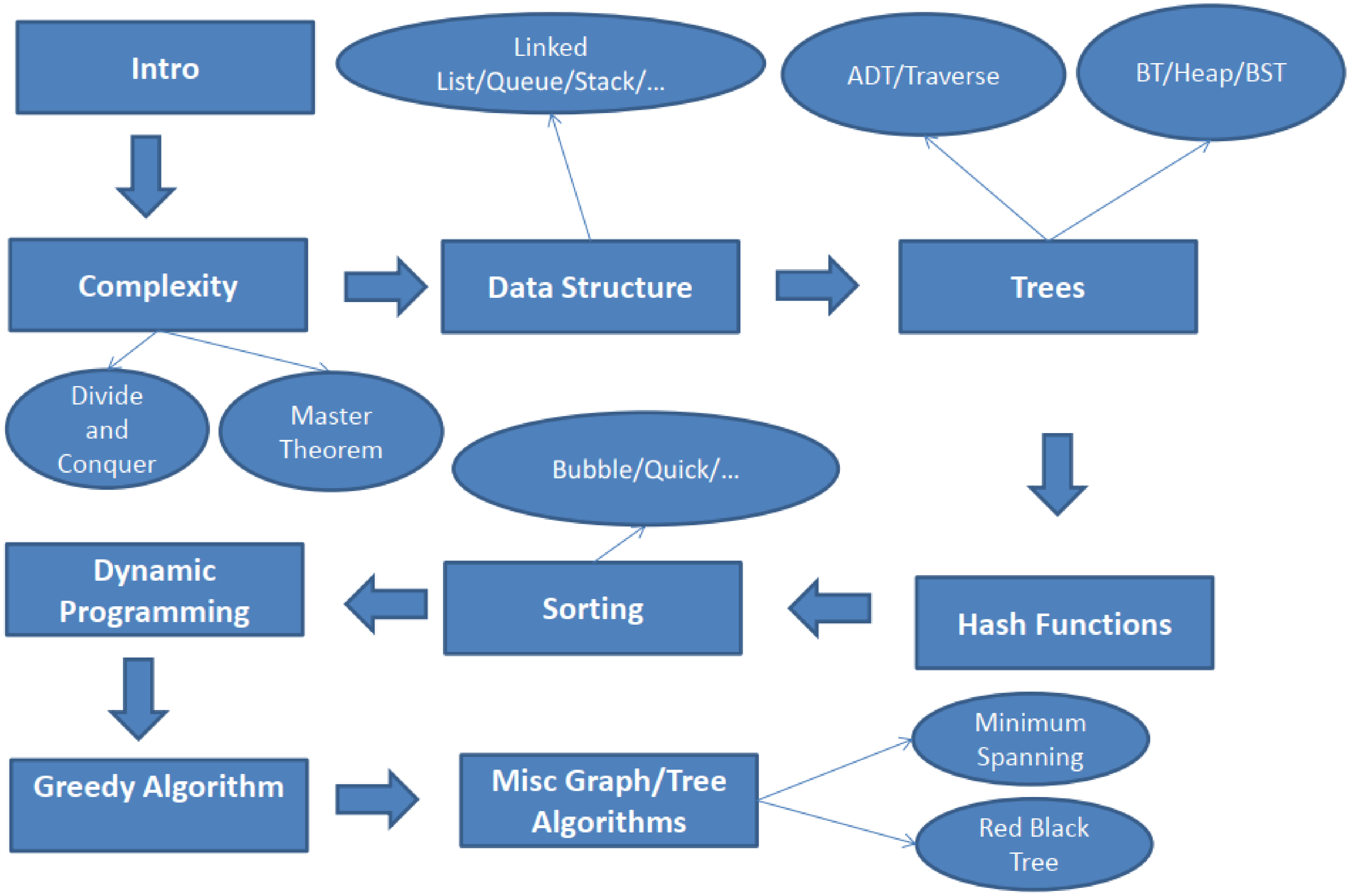
Sorting

Dynamic Programming

Greedy Algorithm

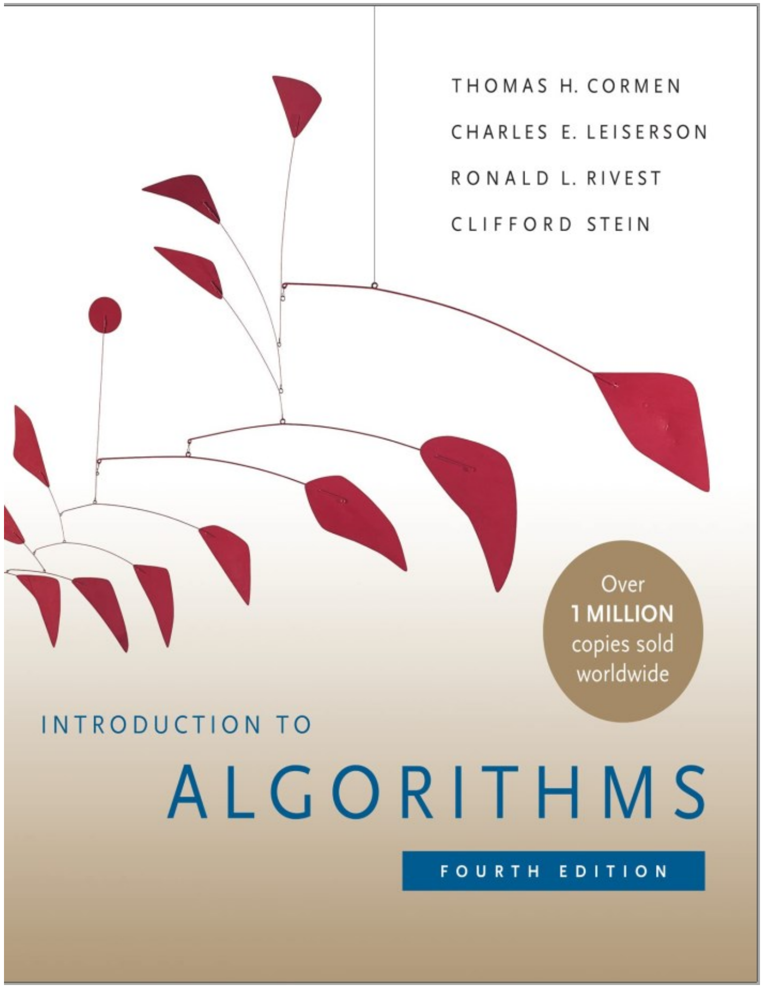
Misc Graph/Tree Algorithms

Our course (big picture)



Materials and References

- (Main) Book:
  - Introduction to Algorithms
    - By: **CLRS**
    - (4th edition)
  - Easy to Read
- Big-Picture is discussed
  - Refer to book for details



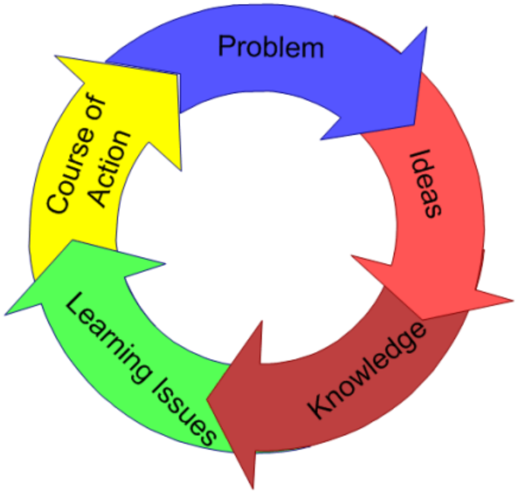
Grading policy

- We are all in the Same Side ☺
  - Exams (12)
  - Exercises (4)
  - Project (3)
  - Class activity (1)
  - Additional activities (1+)
- You must get at least 50% of exams' grades to pass the course.

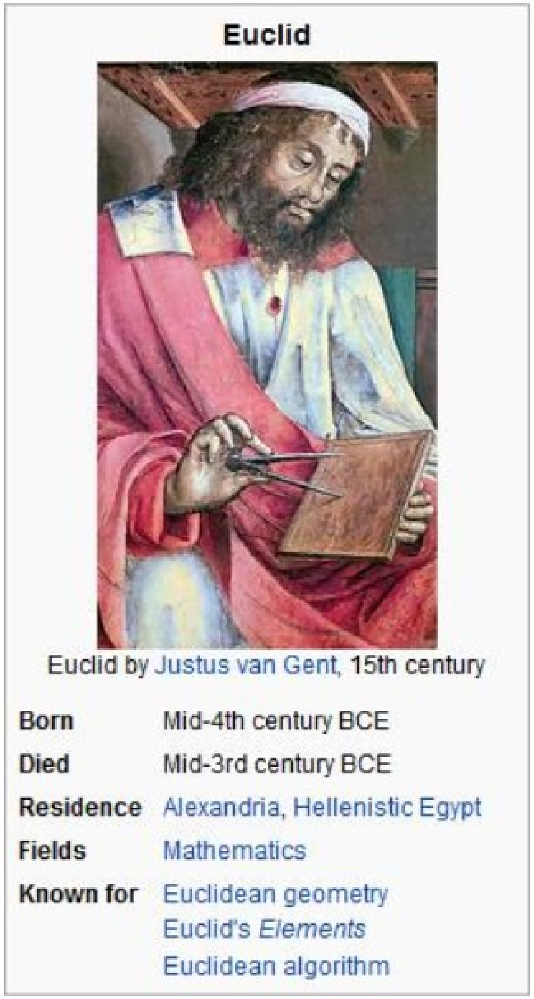
Problem-based Learning

- Traditional Teaching
  - Professor → slides/Blackboard
  - Teaching Assistance → Solve practical
- Problem Based Learning (PBL)
  - More Student-Centric
  - Professor proposes a problem
  - Students explore the solution space
  - Conditions:
    - Number of students

Problem-Based Learning Process



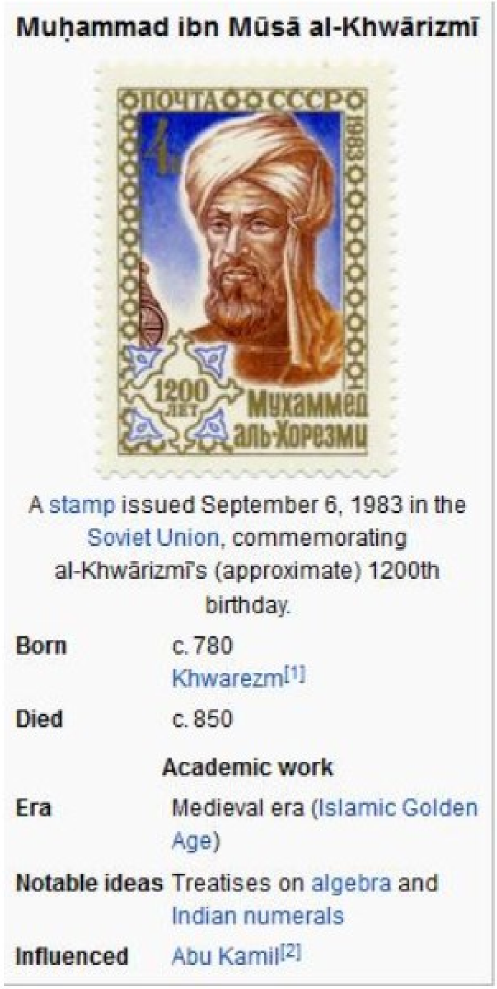
The great thinkers of our field



Euclid

Euclid by Justus van Gent, 15th century

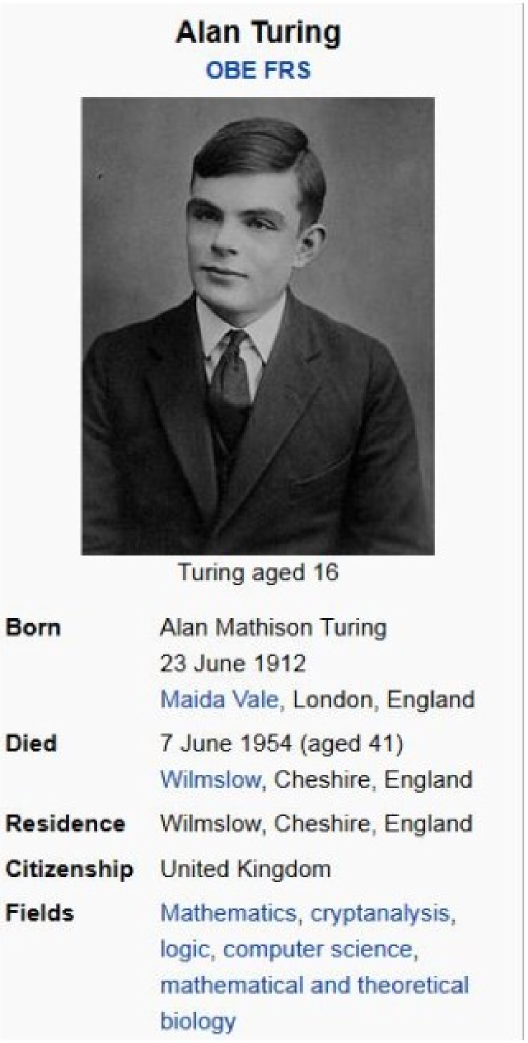
Born	Mid-4th century BCE
Died	Mid-3rd century BCE
Residence	Alexandria, Hellenistic Egypt
Fields	Mathematics
Known for	Euclidean geometry Euclid's *Elements* Euclidean algorithm



Muḥammad ibn Mūsā al-Khwārizmī

A stamp issued September 6, 1983 in the Soviet Union, commemorating al-Khwārizmī's (approximate) 1200th birthday.

Born	c. 780 Khwarezm[1]
Died	c. 850
	Academic work
Era	Medieval era (Islamic Golden Age)
Notable ideas	Treatises on algebra and Indian numerals
Influenced	Abu Kamil[2]



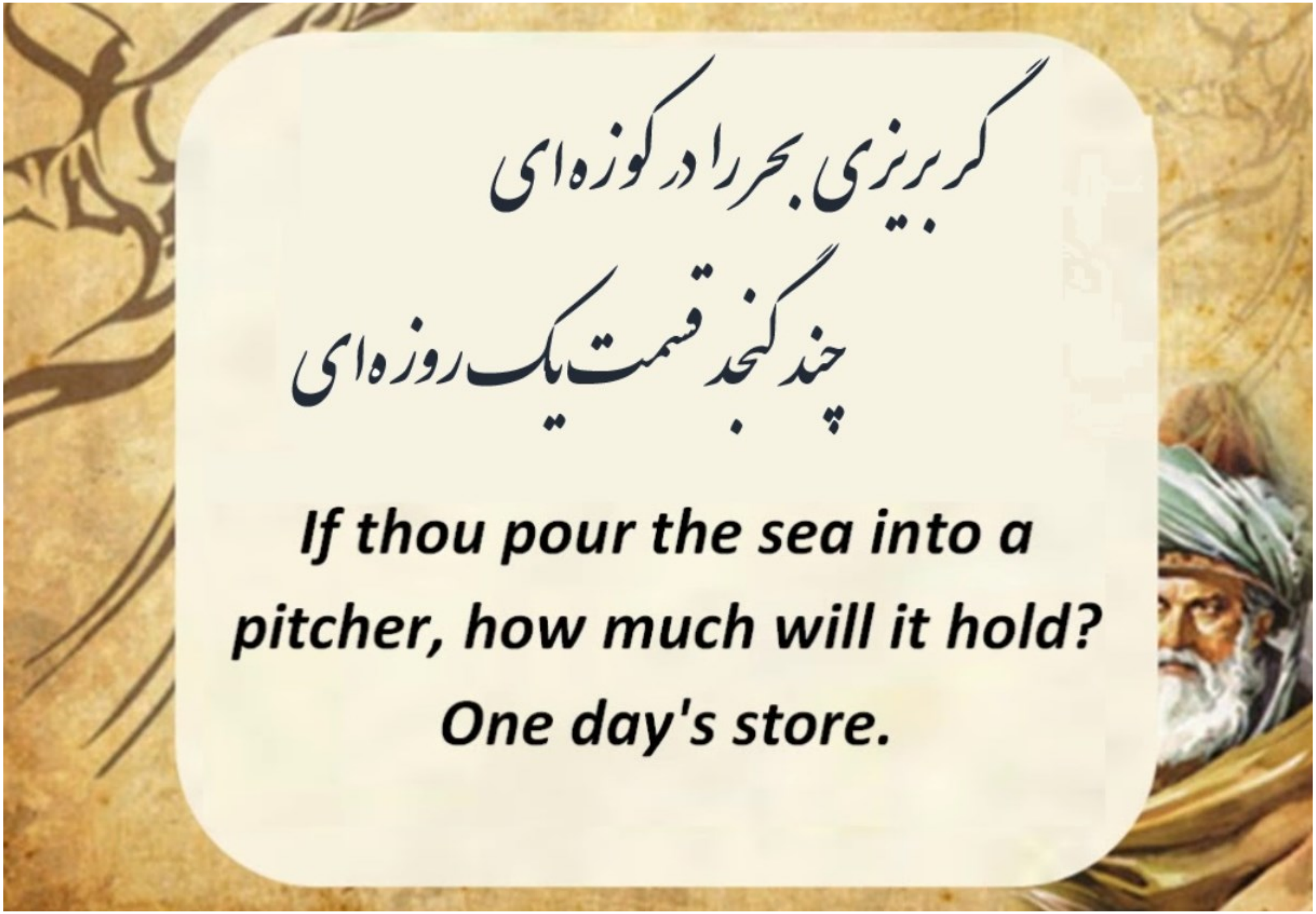
Alan Turing

OBE FRS

Turing aged 16

Born	Alan Mathison Turing 23 June 1912 Maida Vale, London, England
Died	7 June 1954 (aged 41) Wilmslow, Cheshire, England
Residence	Wilmslow, Cheshire, England
Citizenship	United Kingdom
Fields	Mathematics, cryptanalysis, logic, computer science, mathematical and theoretical biology

Stay hungry stay foolish



گر بریزی بحر را در کوزه ای
چند گنجد قسمت یک روزه ای

If thou pour the sea into a
pitcher, how much will it hold?
One day's store.