01 - Introduction to Design and Analysis of Algorithms

[KOMS119602] & [KOMS120403]

Design and Analysis of Algorithm (2021/2022)

Dewi Sintiari

Prodi S1 Ilmu Komputer
Universitas Pendidikan Ganesha

Week 7-11 February 2022

Practical matters

- **Credit:** 3 SKS
- **Lecturer:** Dewi Sintiari
  - email: nld.sintiari@gmail.com
- **Evaluation:**
  - Presence ($\geq 75\%$) + attitude: 20%
  - Quiz / Take-home assignments (theoretical & programming): 40%
  - Midterm exam (written): 20%
  - Final exam (written): 20%
  - Bonus: writing an article, writing in wikipedia?
  - Grade = 20% Presence + 40% Assignments + 20% Midterm + 20% Final + Bonus

What are algorithms and why do we need them?

A simple algorithm:

Recipe of Indomie goreng

Ingredients

Steps

1. .....
2. .....
3. .....
4. .....
5. .....
6. .....
7. .....

Result

A simple algorithm:

Recipe of Teh celup

Ingredients

Characteristics of algorithms

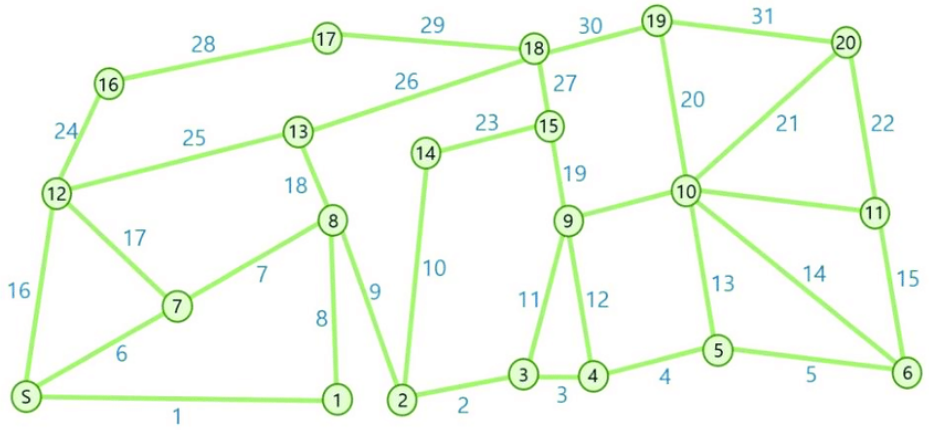
Important components in algorithms:

- Initial situation/condition/input (*Initial state*): can take zero or more inputs.
- Final situation/condition/output (*Output state*): at least one output.
- Definiteness: Each step must be clear, well-defined and precise. There should be no any ambiguity.
- Finiteness: should have finite number of steps and it should end after a finite time.
- Effectiveness: each step must be simple and should take a finite amount of time.
- Constraints given in the beginning and during composing the algorithm (*Constraint and assumption*)

Example of classical algorithmic problems

1. Traveling Salesman Problem (TSP)

Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?



2. Integer Knapsack Problem

Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.

The study of algorithms

Why do we need an algorithm?

1. To understand the basic idea or the flow of the problem.
2. To find an approach to solve the problem. A good design can produce a good solution.
3. To understand the basic principles of designing the algorithms.
4. Compare the performance of the algorithm w.r.t. other techniques.
5. To improve the efficiency of existing techniques.
6. It is the best method of description without describing the implementation detail.
7. To measure the behavior (or performance) of the methods in all cases (best cases, worst cases, average cases)
8. We can measure and analyze the complexity (time and space) of the problems concerning input size without implementing and running it; it will reduce the cost of design.

DAA helps to:

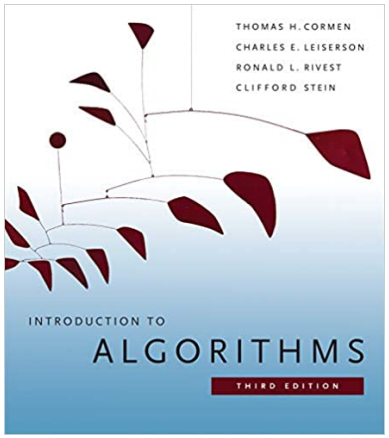
- Design the algorithms for solving problems in CS.

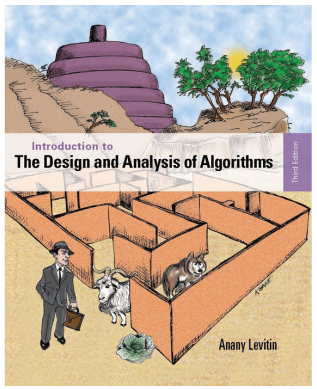
Outline of the semester

1. Introduction to design and analysis of algorithms
2. Complexity analysis of algorithms
3. Brute Force algorithm
4. Greedy algorithm
5. Recursive algorithm
6. Divide-and-Conquer algorithm
7. Decrease-and-Conquer algorithm
8. Transform-and-Conquer algorithm
9. BFS and DFS algorithms
10. Backtracking algorithm
11. Branch & Bound
12. Dynamic programming
13. Sorting algorithms
14. Graph algorithms
15. Computational complexity theory (P, NP, NP-C)

References

- Introduction to Algorithms (Thomas H. Cormen, C. E. Leiserson, R. Rivest, C. Stein), The MIT Press, 1989.
- Introduction to the Design and Analysis of Algorithms (Anany Levitin), Pearson, 2012.





- Kuliah pengantar Strategi Algoritma (Rinaldi Munir ITB)
- e-Modul Struktur Data dan Analisis Algoritma (Made Windu A. Kesiman, PTI Undiksha)

2. Complexity analysis

Determining a functions of time complexity and space complexity. An algorithm is said to be *efficient* when this function's values are small, or grow slowly compared to the growth in the size of input.

- n: the size of input
- $T(n)$: the number of computations/steps (comparison, arithmetic operations, accessing an array, etc.)
- $S(n)$: memory/storage space

- Measuring complexity: best-case, worst-case, and average-case
- We usually estimate the complexity *asymptotically*, i.e., to estimate the complexity function for arbitrarily large input. We use Big O (*upper-bound*), Big-omega (*lower-bound*) and Big-theta (*tight-bound*) notations.

3. Brute Force algorithm / Exhaustive search

This is the most basic and simplest type of algorithm. It systematically enumerate all possible candidates for the solution

Classification of algorithms based on the strategy

1. Direct solution: brute-force, greedy
2. Space-state base: backtracking, branch and bound
3. Top-down solution: divide-and-conquer, decrease-and-conquer, transform-an-conquer, dynamic programming, BFS & DFS
4. Bottom-up solution: dynamic programming