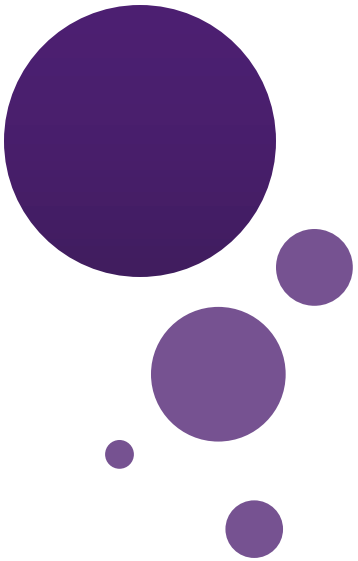


Lecture 1: Introduction to Discrete Structures

Dr. Chengjiang Long
Computer Vision Researcher at Kitware Inc.
Adjunct Professor at SUNY at Albany.
Email: **clong2@albany.edu**

Self-introduction

2009.6 B.S. Wuhan, China

2011.6 M.S. Wuhan, China

2013.5-8 Research Intern, NEC Laboratories America, Cupertion, CA

2015.5-8 Research Intern, GE Global Research, Niskayuna, NY

2015.10 Ph.D. Hoboken, NJ

2016.2-now Computer Vision Researcher, Senior R&D Engineer, Kitware, Clifton Park, NY

2018.1-5 Adjunct Professor, Rensselaer, Troy, NY

2018.8-now Adjunct Professor, University at Albany, State University of New York, Albany, NY





Outline

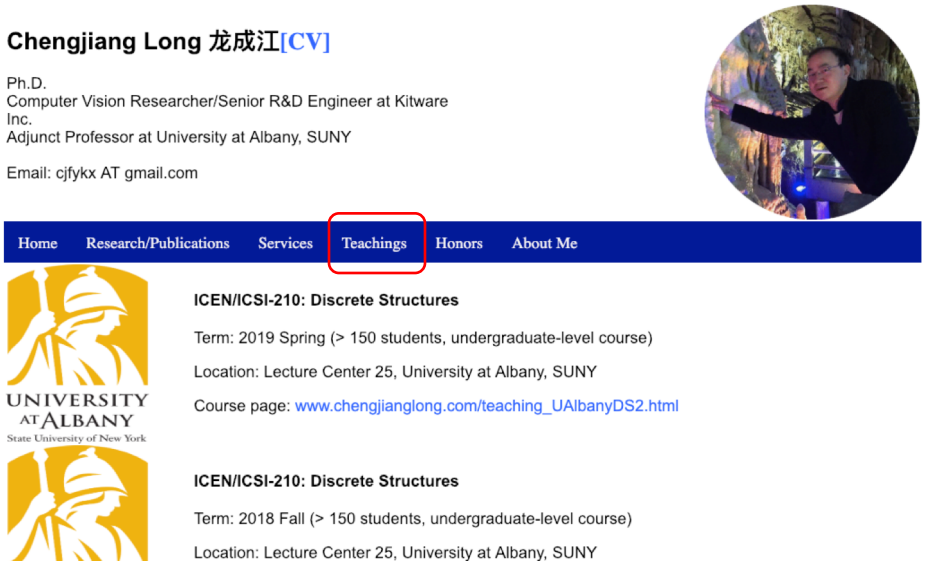
- Course Information
- What is Discrete Structures?
- Why Mathematics?
- Steps to Solve a Problem

Outline

- **Course Information**
- What is Discrete Structures?
- Why Mathematics?
- Steps to Solve a Problem

Course information

- **ICEN/ICSI-210** Discrete Structures
- **Term**: Spring 2019
- **Instructor**: Dr. Chengjiang Long
- **Email:** clong2@albany.edu
- **Class time**: 11:30 am—12:25 pm, Monday, Wednesday & Friday
- **Location**: LC 25
- **Teaching Assistant**: Siqian Zhao (szhao2@albany.edu) and Ronke Osipitan (iosipitan@albany.edu)
- **Student Assistant**: Mehrdad Mirzaei (mmirzaei@albany.edu) and Alfred, Shalin (salfred@albany.edu)
- **Course Website**: www.chengjianglong.com/teaching_UAlbanyDS2.html



Chengjiang Long 龙成江[CV]

Ph.D.
Computer Vision Researcher/Senior R&D Engineer at Kitware Inc.
Adjunct Professor at University at Albany, SUNY

Email: cjfykx AT gmail.com

Home | Research/Publications | Services | Teachings | Honors | About Me

UNIVERSITY AT ALBANY
State University of New York

ICEN/ICSI-210: Discrete Structures

Term: 2019 Spring (> 150 students, undergraduate-level course)

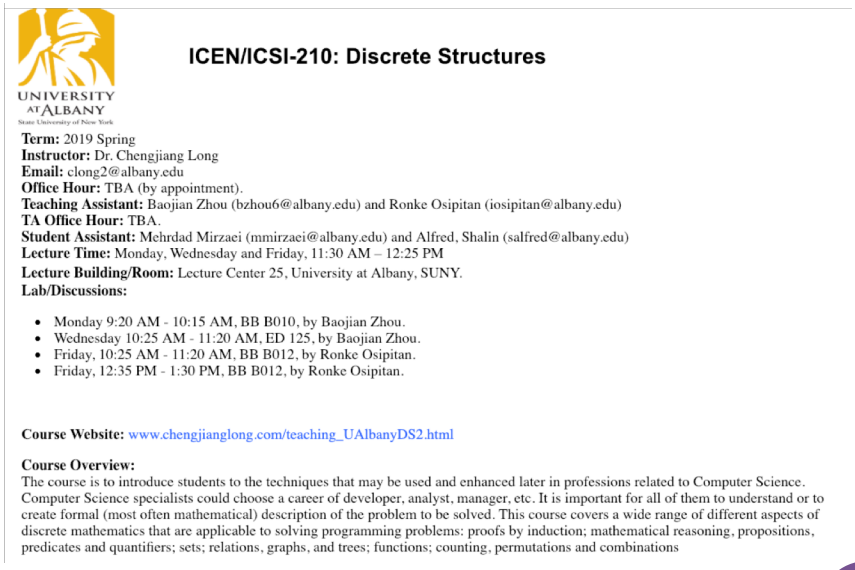
Location: Lecture Center 25, University at Albany, SUNY

Course page: www.chengjianglong.com/teaching_UAlbanyDS2.html

ICEN/ICSI-210: Discrete Structures

Term: 2018 Fall (> 150 students, undergraduate-level course)

Location: Lecture Center 25, University at Albany, SUNY



UNIVERSITY AT ALBANY
State University of New York

ICEN/ICSI-210: Discrete Structures

Term: 2019 Spring
Instructor: Dr. Chengjiang Long
Email: clong2@albany.edu
Office Hour: TBA (by appointment).
Teaching Assistant: Baojian Zhou (bzhou6@albany.edu) and Ronke Osipitan (iosipitan@albany.edu)
TA Office Hour: TBA.
Student Assistant: Mehrdad Mirzaei (mmirzaei@albany.edu) and Alfred, Shalin (salfred@albany.edu)
Lecture Time: Monday, Wednesday and Friday, 11:30 AM – 12:25 PM
Lecture Building/Room: Lecture Center 25, University at Albany, SUNY.
Lab/Discussions:

- Monday 9:20 AM - 10:15 AM, BB B010, by Baojian Zhou.
- Wednesday 10:25 AM - 11:20 AM, ED 125, by Baojian Zhou.
- Friday, 10:25 AM - 11:20 AM, BB B012, by Ronke Osipitan.
- Friday, 12:35 PM - 1:30 PM, BB B012, by Ronke Osipitan.

Course Website: www.chengjianglong.com/teaching_UAlbanyDS2.html

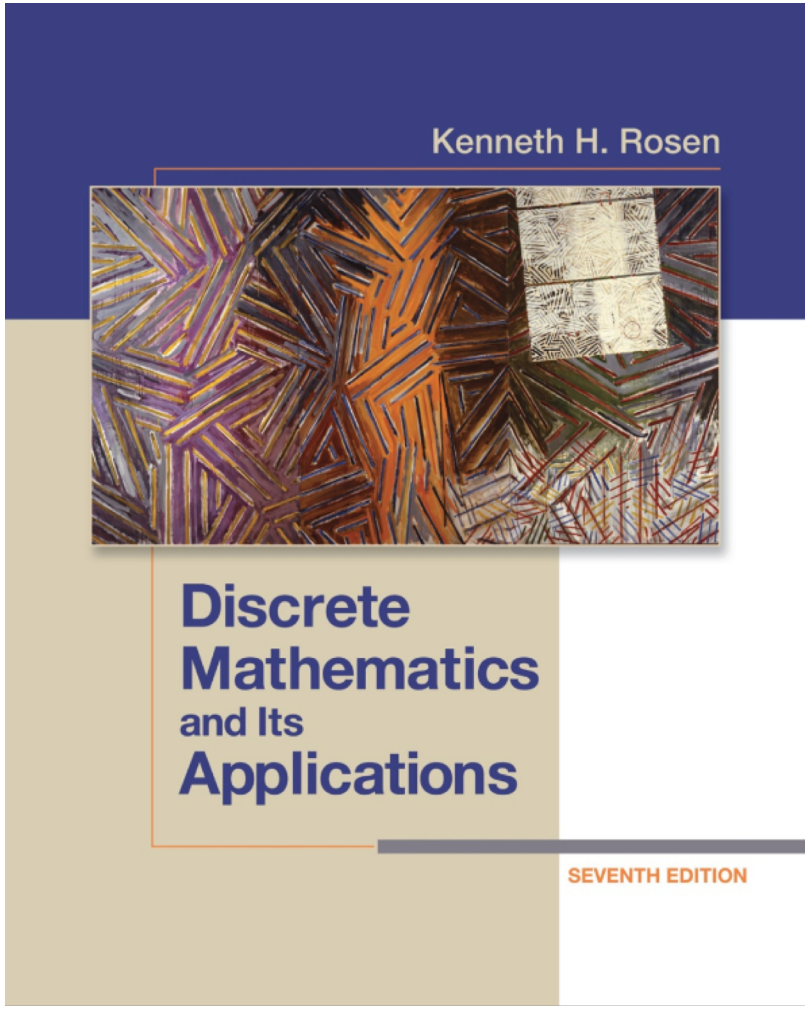
Course Overview:
The course is to introduce students to the techniques that may be used and enhanced later in professions related to Computer Science. Computer Science specialists could choose a career of developer, analyst, manager, etc. It is important for all of them to understand or to create formal (most often mathematical) description of the problem to be solved. This course covers a wide range of different aspects of discrete mathematics that are applicable to solving programming problems: proofs by induction; mathematical reasoning, propositions, predicates and quantifiers; sets; relations, graphs, and trees; functions; counting, permutations and combinations

Topics and textbooks

Topics covering:

- Logic
- Proof
- Sets
- Functions
- Counting
- Discrete probability
- Relations
- Graph
- Tree

This textbook has been used in over 500 institutions around the world.



http://www2.fiit.stuba.sk/~kvasnicka/Mathematics%20for%20Informatics/Rosen_Discrete_Mathematics_and_Its_Applications_7th_Edition.pdf

Prerequisites

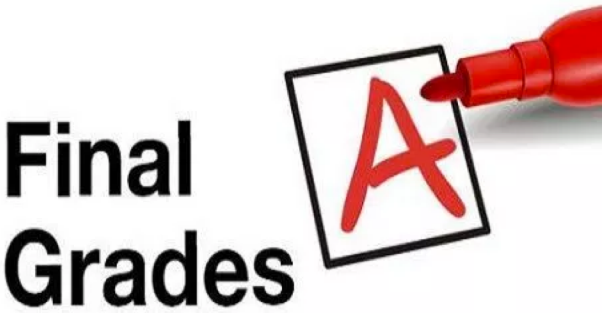
- Students should have a fundamental understanding of mathematical reasoning as well as be competent in solving applied algebra problems.
- The most important prerequisites are interest in the subject, willingness to dedicate necessary resources in terms of time and intellectual effort, and willingness to actively participate in the learning process.
- **No programming skills are required to pass the course!**

Objectives of This Course

- The goal of the course is to introduce students to the techniques that may be used and enhanced later in professions related to Computer Science.
  - To learn basic mathematical concepts, e.g. sets, functions, graphs
  - To be familiar with formal mathematical reasoning, e.g. logic, proofs
  - To improve problem solving skills
  - To see the connections between discrete mathematics and computer science

Grading

- Lecture, lab & discussion participation: 5%
- 12 Homeworks (almost every Monday) : 50%
- 2 Mid-term exams: 15%
- Final exam: 30%
- Final grade: A(≥92), A-(≥90), B+(≥87), B(≥82), B-(≥80), C+(≥77), C(≥72), C-(≥70), and F (< 70).



Optional Problems and Attendance Bonus

- Extra points: 20% for each homework. Note that this is optional, the purposes of this design is to encourage the self-motivated students to challenge themselves and give them more chances to get a higher score.

- Attendance bonus: I would like to give the bonus to reward those students whose attendance is less than 3. For those who never miss any class, I will give them 5 extra points on the final grade. For those who miss only 1 class, I will give them 2 extra points. And for those who miss 2 classes, I will give them 1 extra point on the final grade.

Schedule

Class	Date	Topic	Reading	Homework	Slides
1	1/23/2019	Introduction			Lecture_1
2	1/25/2019	Propositional Logic	Ch 1.1-1.2		Lecture_2
3	1/28/2019	Predicate Calculus and Quantifiers	Ch 1.3-1.4	Homework_1	Lecture_3
4	1/30/2019	Quantifiers and Rules of Reference	Ch 1.4-1.5		Lecture_4
5	2/1/2019	Proofs	Ch 1.5-1.8		Lecture_5
6	2/4/2019	Introduction to Sets	Ch 2.1	Homework_2	Lecture_6
7	2/6/2019	Set Operations and Introduction to Functions	Ch 2.2-2.3		Lecture_7
8	2/8/2019	Functions	Ch 2.3		Lecture_8
9	2/11/2019	Sequences and Summation (1)	Ch 2.4	Homework_3	Lecture_9
10	2/13/2019	Sequences and Summation (2)	Ch 2.4		Lecture_10
11	2/15/2019	Cardinality of Sets	Ch 2.5		Lecture_11
12	2/18/2019	Matrices	Ch 2.6	Homework_4	Lecture_12
13	2/20/2019	Algorithms	Ch 3.1		Lecture_13
14	2/22/2019	Growth of Functions and Complexity	Ch 3.2-3.3		Lecture_14
15	2/25/2019	Integers and Division	Ch 4.1	Homework_5	Lecture_15
16	2/27/2019	Midterm Exam 1	Ch 1.1-1.8, Ch 2.1-2.6		
17	3/1/2019	Modular Arithmetic	Ch 4.1		Lecture_16
18	3/4/2019	Integer Representations	Ch 4.2	Homework_6	Lecture_17
19	3/6/2019	Primes and Greatest Common Divisors	Ch 4.3		Lecture_18
20	3/8/2019	Cryptograph	Ch 4.6		Lecture_19
21	3/11/2019	TBA		Homework_7	
22	3/13/2019	Mathematical Induction	Ch 5.1-5.2		Lecture_20
23	3/15/2019	Strong Induction and Well-Ordering	Ch 5.2		Lecture_21
	3/18/2019	Class Suspended -- Spring Break			
	3/20/2019	Class Suspended -- Spring Break			
	3/22/2019	Class Suspended -- Spring Break			
24	3/25/2019	Structural Recursion and Induction	Ch 5.3	Homework_8	Lecture_22
25	3/27/2019	Recursive Algorithms	Ch 5.4		Lecture_23
26	3/29/2019	Recursive Algorithms and Basic Counting Rules	Ch 5.4 and Ch 6.1		Lecture_24
27	4/1/2019	Pigeonhole Principle, Permutations and Combination	Ch 6.2-6.3	Homework_9	Lecture_25
28	4/3/2019	Binomial Coefficients and Identities	Ch 6.4		Lecture_26
29	4/5/2019	Inclusion-exclusion Principle	Ch 6.4		Lecture_27
30	4/8/2019	Discrete Probability	Ch 7.1-7.2	Homework_10	Lecture_28
31	4/10/2019	Conditional Probability	Ch 7.2-7.3		Lecture_29
32	4/12/2019	Midterm Exam 2	Ch 3.1 - Ch 6.4		
33	4/15/2019	TBA		Homework_11	
34	4/17/2019	Bayes Rules, Expected Value, Variance and Binorminal Distribution	Ch 7.3-7.4		Lecture_30
35	4/19/2019	Relations (1)	Ch 9.1-9.3		Lecture_31
	4/22/2019	Class Suspended -- Easter			
36	4/24/2019	Relations (2)	Ch 9.4-9.5		Lecture_32
37	4/26/2019	Graph Theory (1)	Ch 10.1-10.5		Lecture_33
38	4/29/2019	Graph Theory (2)	Ch 10.1-10.5	Homework_12	Lecture_34
39	5/1/2019	Trees	Ch 11.1-11.5		Lecture_35
40	5/3/2019	Recap and Review (1)			
41	5/6/2019	Recap and Review (2)			
42	5/8/2019	Recap and Review (3)			
43	5/13/2019 (3:30pm – 5:30pm)	Final Exam	Ch 1-Ch 11	Final Exam	

Note: The above course schedule may be subject to change. Please do check the latest update.

Note: this may be subject to change.

Rules

- **Need to be absent from class?**
  - 1 point per class: please send notification and justification at least 2 days before the class.
- **Late submission of homework?**
  - The maximum grade you can get from your late homework decreases 25% per day.
  - **> 3 hours after the deadline** will be considered as “late submission”.
- **Zero tolerance on plagiarism**!!!
  - The first time you receive zero grade for the assignment.
  - The second time you get “F” in your final grade.
  - Refer to the University at Albany, SUNY's honor system for your behavior.

Outline

- Course Information
- **What is Discrete Structures?**
- Why Mathematics?
- Steps to Solve a Problem

Problem Solving Requires Mathematical Rigor

- Your boss is not going to ask you to solve
  - an MST (Minimal Spanning Tree) or
  - a TSP (Travelling Salesperson Problem)
- Rarely will you encounter a problem in an abstract setting
- However, he/she may ask you to build a rotation of the company's delivery trucks to minimize fuel usage
- It is up to you to determine
  - a proper model for representing the problem and
  - a correct or efficient algorithm for solving it

Scenario I

- A limo company has hired you/your company to write a computer program to automate the following tasks for a large event.

- **Task1**: In the first scenario, businesses request
  - limos and drivers
  - for a fixed period of time, specifying a start data/time and end date/time and
  - a flat charge rate
- The program must generate a schedule that accommodates the maximum number of customers

Scenario II

- **Task 2**: In the second scenario, the limo service allows customers to bid on a ride so that the highest bidder get a limo when there aren't enough limos available

- The program should make a schedule that is feasible (no limo is assigned to two or more customers at the same time) while maximizing the total profit

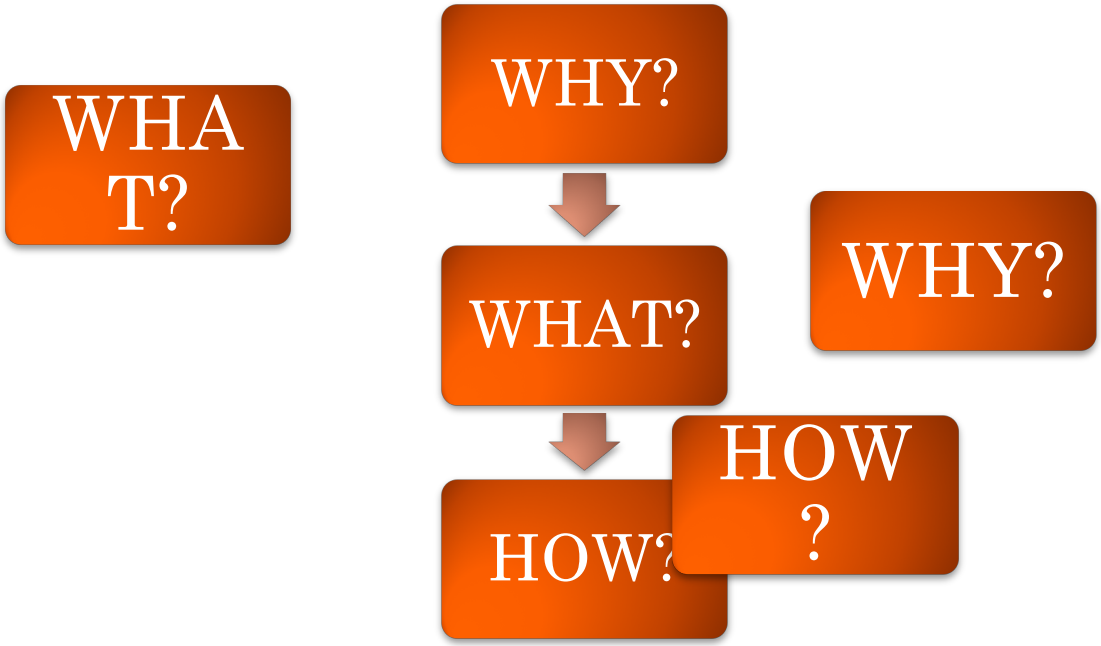
Scenario III

- **Task 3**: Here each customer is allowed to specify a set of various times and bid an amount for the entire event.
- The limo service must choose to accept the entire set of times or reject it
- The program must again maximize the profit.

What's your job?

- Build a mathematical model for each scenario
- Develop an algorithm for solving each task
- Justify that your solutions work
  a) Prove that your algorithms terminate. Termination
  b) Prove that your algorithms find a solution when there is one. Completeness
  c) Prove that the solution of your algorithms is correct Soundness
  d) Prove that your algorithms find the best solution (i.e., maximize profit). Optimality (of the solution)
  e) Prove that your algorithms finish before the end of life on earth. Efficiency, time & space complexity

Very Important Questions



Computer Science vs. Programming

Why to do –

Your boss

What to do (what can be done) –

Computer Science

How to do –

Software Engineering

Discrete Structures is one of the disciplines studied by Computer Science students.

Discrete structures are systems and objects studied in **discrete mathematics**.

Outline

- Course Information
- What is Discrete Structure?
- **Why Mathematics?**
- Steps to Solve a Problem

Why Mathematics?

- Computers use discrete structures to represent and manipulate data.
- Discrete structures are the basic building block for becoming a Computer Scientist
- Computer Science is not Programming
- Computer Science is not Software Engineering
- Edsger Dijkstra: “Computer Science is no more about computers than Astronomy is about telescopes.”
- Computer Science is about **problem solving**.

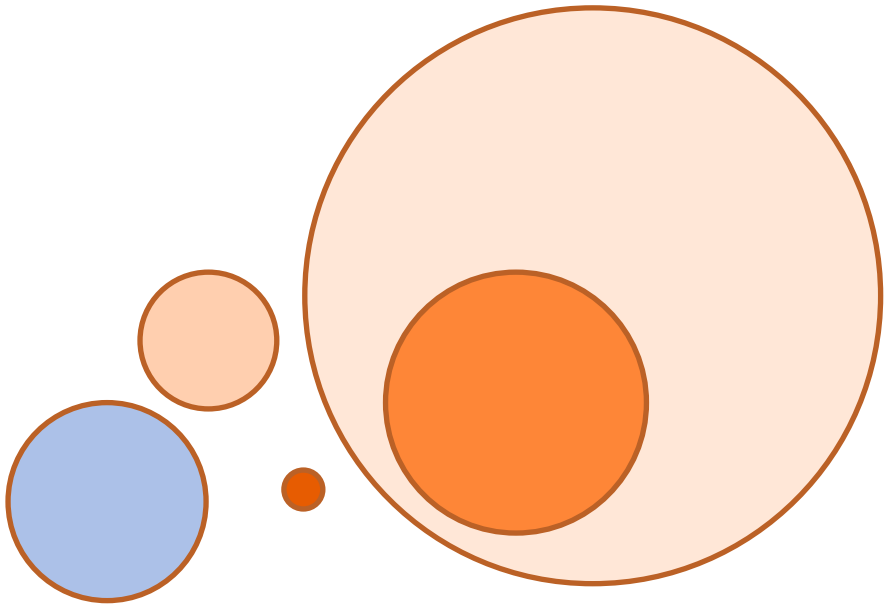
Why Mathematics?

- Mathematics is the tool that arms practitioners with form (theoretical) approaches to problem solving.

- Formal approach allows:
  - to manage very small and very large numbers.
  - to resue solutions.

Example:

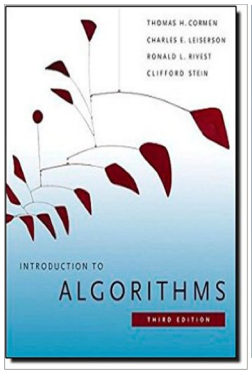
Area of a circle $A = \pi \times r^2$

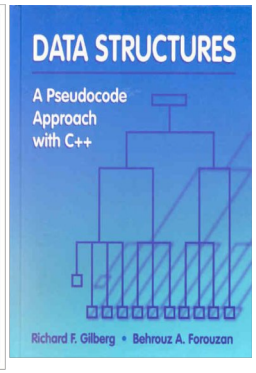


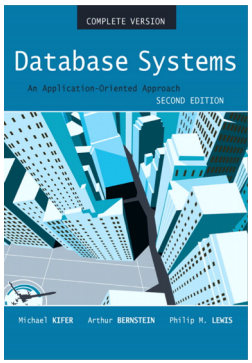
Why Mathematics?

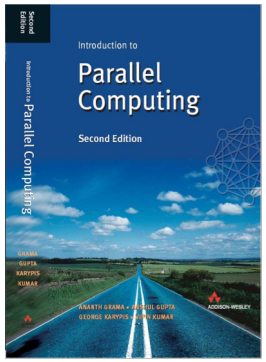
Design efficient computer systems.

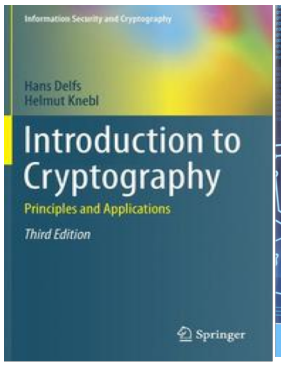
- How did Google manage to build a fast search engine?
- What is the foundation of internet security?

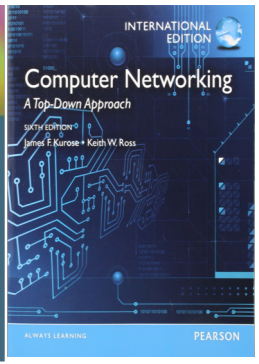












...

algorithms, data structures, database, parallel computing, distributed systems, cryptography, computer networks…

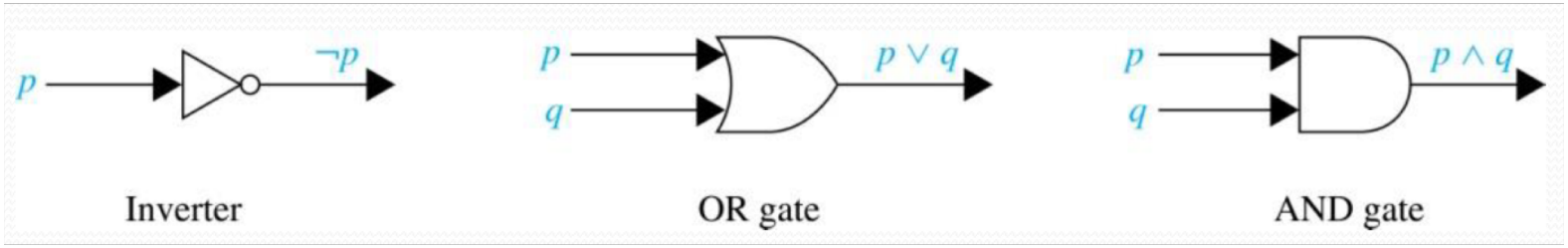
Logic, number theory, counting, graph theory…

Topic 1: Logic and Proofs

How do computers think?

Logic: propositional logic, first order logic

Proof: induction, contradiction



$$\frac{x_1 + x_2 + \ldots + x_n}{n} \geq \sqrt[n]{x1 \cdot x_2 \cdots x_n}$$

Artificial intelligence, database, circuit, algorithms

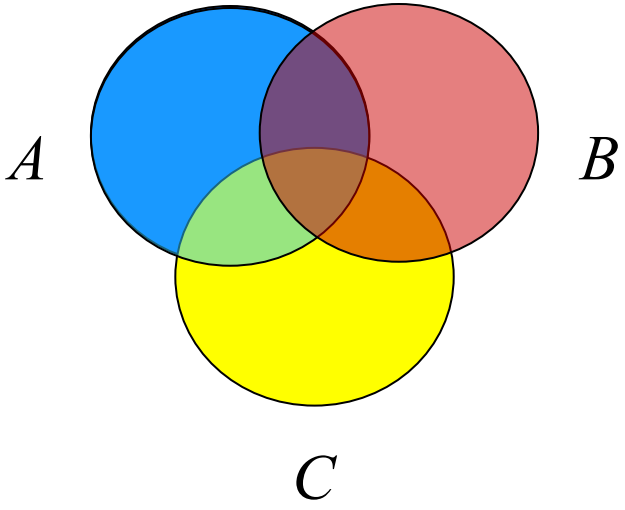
Topic 2: Number Theory

- Number sequence
- Euclidean algorithm
- Prime number, modular arithmetic, Chinese remainder theorem
- Cryptography, RSA protocol

Cryptography, coding theory, data structures

Topic 3: Counting

- Sets and Functions
- Combinations, Permutations, Binomial theorem
- Counting by mapping, pigeonhole principle
- Recursions



Probability, algorithms, data structures

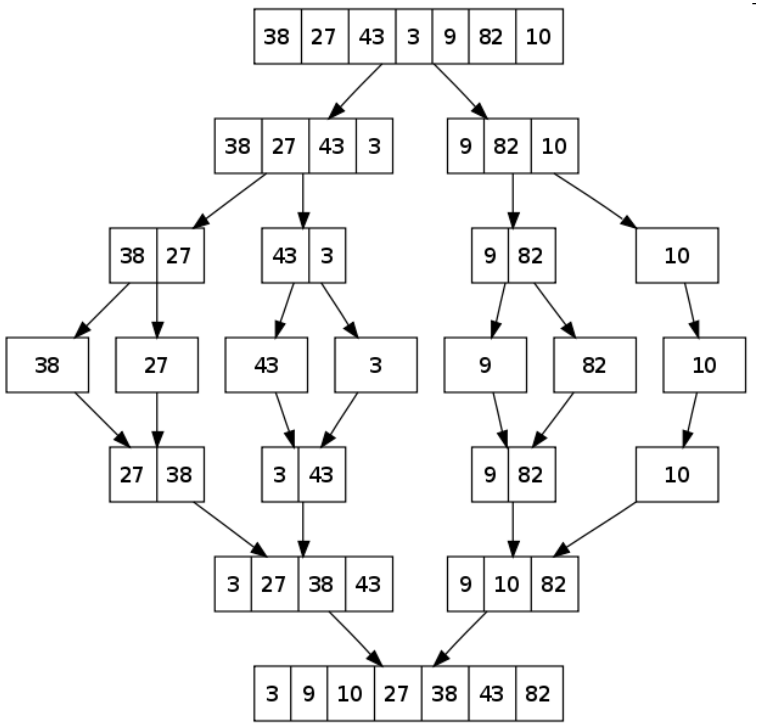
Topic 3: Counting

How many steps are needed to sort n numbers?

Algorithm 1 (Bubble Sort):

Every iteration moves the i-th smallest number to the i-th position

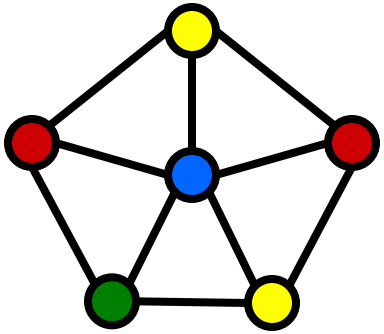
Algorithm 2 (Merge Sort):

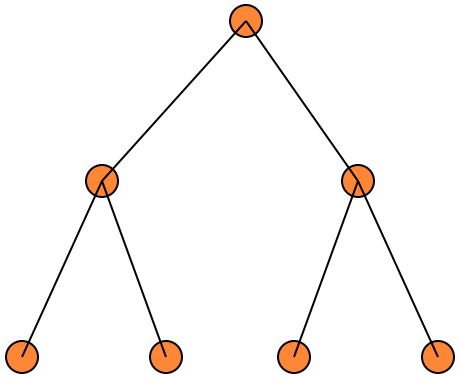


Which algorithm runs faster?

Topic 4: Graph Theory

- Graphs, Relations
- Degree sequence, Eulerian graphs, isomorphism
- Trees
- Matching
- Coloring

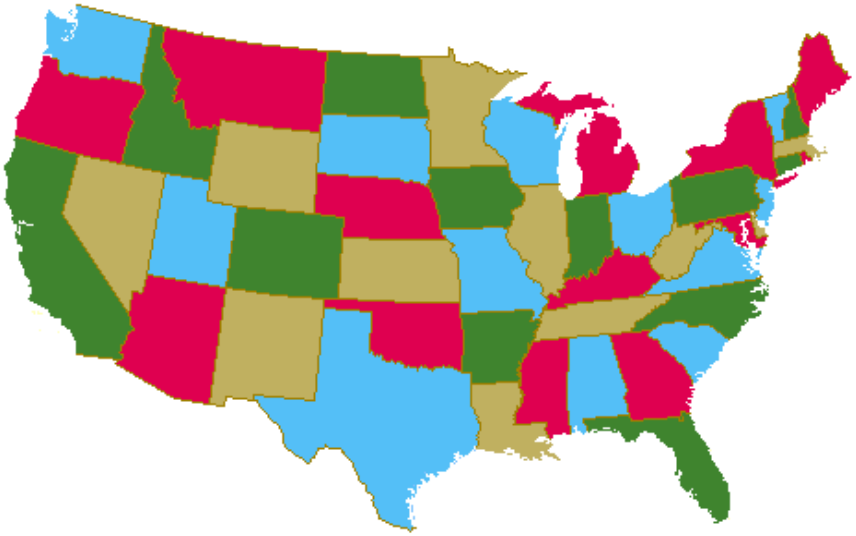


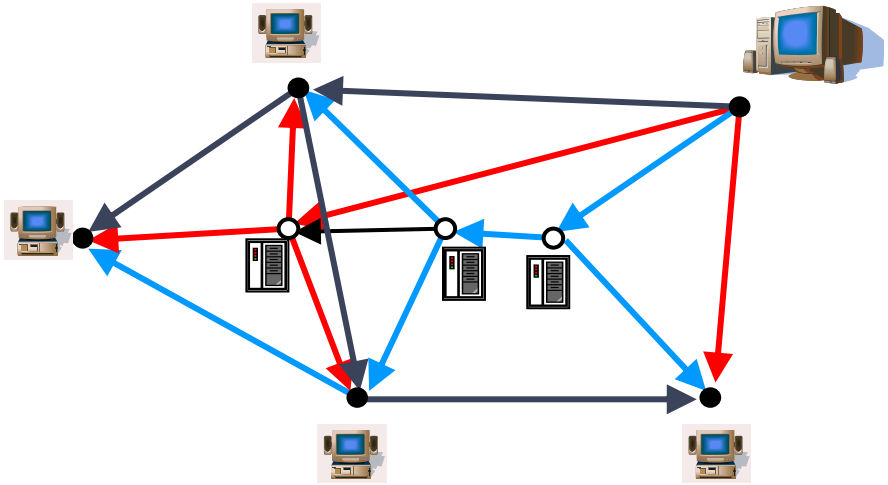


Computer networks, circuit design, data structures

Topic 4: Graph Theory

How to color a map?





How to send data efficiently?

The subjects covered in this course

- **Propositional logic** – language, essence and rules of reasoning using propositions.
- **Predicate logic** – reasoning with statements involving variables.
- **Sets** – the basis of every theory in computer science.
- **Functions** – the special type of relations, one of the most important concepts in data structures, formal languages and automata, and analysis of algorithms.
- **Mathematical reasoning** – recursive definitions and mathematical induction.
- **Relations** – one of the key concepts in many subjects on computer and computation. For example, a database is viewed as a set of relations and database query languages are constructed based on operations on relations and sets.
- **Graphs** – good example of discrete structures and one of the most useful models for computer scientists and engineers in solving problems.
- **Trees** – introduction to trees, tree traverse, spanning tree.

Outline

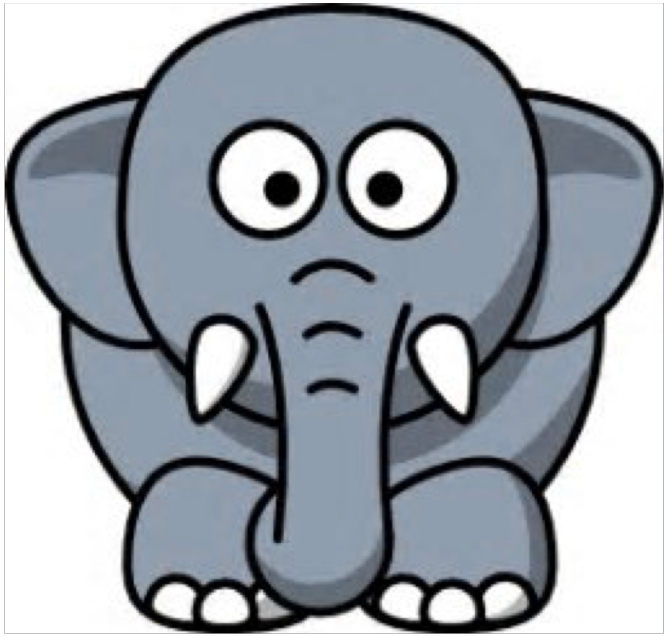
- Course Information
- What is Discrete Structures?
- Why Mathematics?
- **Steps to Solve a Problem**

Problems in Computer Science Field

- How many possible routes could be organized between computers inside the network?
- What time is required to sort a list of objects?
- What is the probability of winning a lottery?
- What is the shortest path between nodes in a computer network?
- .....

Four Steps to Solve a Problem

1. Identify the problem
2. Design solution plan
3. Implement the plan
4. Verify the solution



Step 1. Identifying the Problem ("WHAT")

- Extract the key parts of the problem
  - find-problems:
    - unknowns,
    - data,
    - conditions,
  - proof-problems:
    - hypothesis,
    - conclusion.
- Consider hidden assumptions, data and conditions.
- Clarify all unknown terms.
- Construct several examples to illustrate what the problem is about.

Step 2. Designing of a Plan ("HOW")

- 2.1. Once you know "WHAT" look for the familiar words and concepts. Often, previously acquired knowledge may give a lead.
- 2.2. If the problem is not similar to one that has been solved previously, look at the problem from different points of view:
  - find facts that are related to the problem; relevant facts usually involve words that are the same as or similar to those in the given problem,
  - look for a helpful idea that shows the way to the end; even an incomplete idea should be considered. Go along with it to a new situation, and repeat this process.

Example 1. The Problem

- A survey of TV viewers shows the following results:
  - To the question "Do you watch comedies?" 352 answered "Yes"
  - To the question "Do you watch sports?" 277 answered "Yes" and
  - To the question "Do you watch both comedies and sports?" 129 answered "Yes"

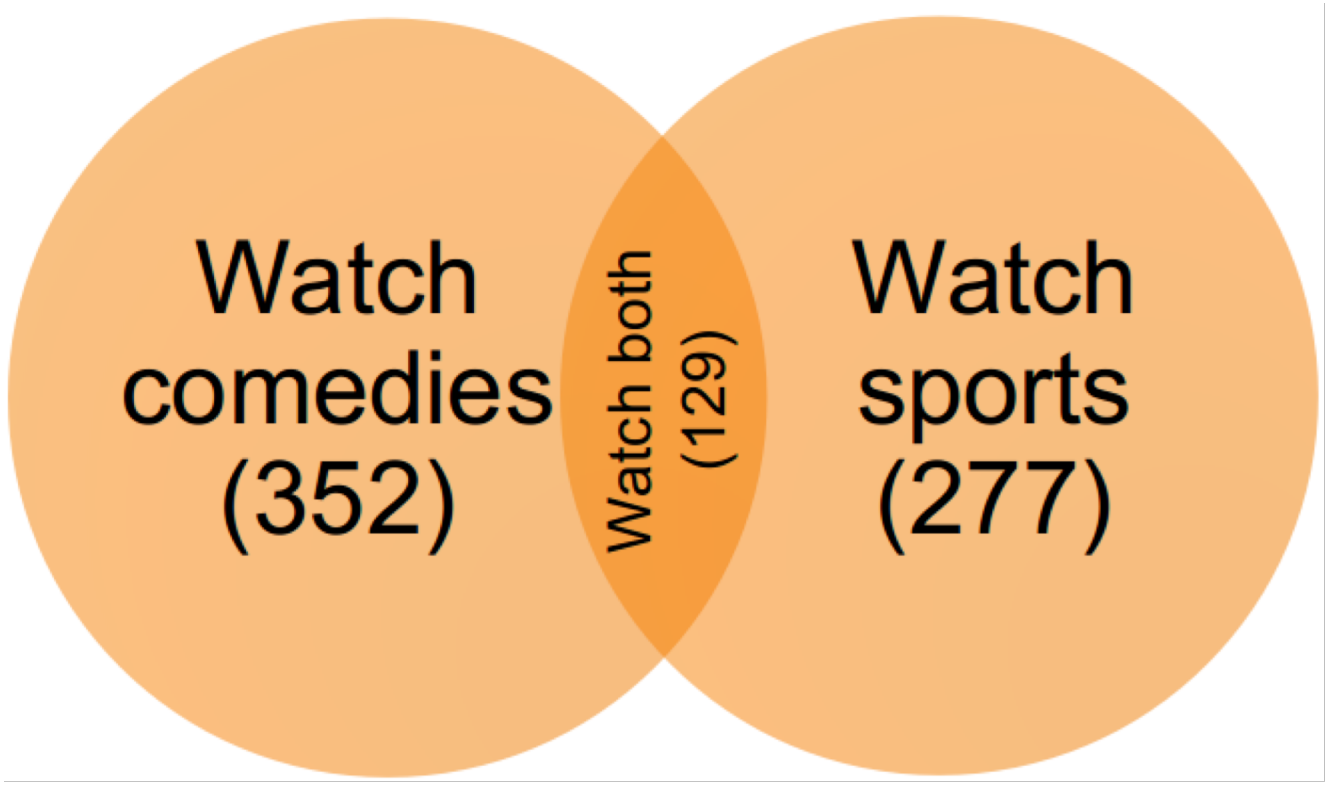
- Given these information, find the share of people
  - who watch *only* comedies,
  - *only* sports,
  - and *both* comedies and sports among people who watch *at least* one of comedies and sports.

Step 1 - Identifying the Problem

- This is a find-type problem, so the key parts are *unknowns*, *data* and *conditions*.
  - The **unknowns** are (a) the percentage of people who watch only comedies, (b) the percentage of people who watch only sports, and (c) the percentage of people who watch both comedies and sports.
  - The **data** are the three numbers: 352, 277 and 129, representing the number of people who watch comedies, sports, and both comedies and sports, respectively. Note that 352 includes people who watch both comedies and sports as well as people who watch only comedies. Similarly for 277.
  - The **conditions** are not explicitly given in the problem statement. But one can see that the percentages must add up to 100, and they must be nonnegative.

Step 1 - Identifying the Problem

- It is always good to have a graphical illustration:



Step 2 - Designing a Plan

- What do we know already? How to compute percentage!
- To compute percentage we need total number and number that represents a share.
- Let's go to the formal description:
  - ***T*** is the total number of people who watch at least one of comedies and sports;
  - ***C*** is the number of people who watch only comedies;
  - ***S*** is the number of people who watch only sports.
- Look at the diagram on the previous slide to find the areas that correspond to ***T***, ***C***, and ***S*** respectively.

Step 2 - Designing a Plan, cont'd

- Thus we have the following relationships among the unknowns:
-

  $C + 129 = 352$
  $S + 129 = 277$
  $C + S + 129 = T$

- Hence, $C = 223$, $S = 148$, and $T = 500$.
- Thus the required percentages are **44.6%**, **29.6%**, and **25.8%**, respectively.

Extrapolation

- Once you have learned how to solve the problem, you have enhanced your problem solving skills. The ability to apply your skills and knowledge to the new problem is important part of critical thinking and is often called "ability to extrapolate."
- It includes such skills as:
  - how to find the similarity between the problems,
  - how to find the difference between the problems,
  - how to see the pattern,
  - how to apply previously acquired knowledge.

Some Helpful Tips

- ✓ (1) Check if you **seen something similar before?**
- ✓ (2) Do a little **analysis** on relationships among data, conditions and unknowns, or between hypothesis and conclusion.
- ✓ (3) What facts do I know **related** to the problem on hand? These are facts on the subjects appearing in the problem. They often involve the same or similar words.
- ✓ (4) Make sure that you know the meaning of the all terms.
- ✓ (5) Compose a **wish list** of intermediate goals and try to reach them.
- ✓ (6) Have you used all the **conditions/hypotheses**? When you are looking for paths to a solution or trying to verify your solution, it is often a good idea to check whether or not you have used all the data/hypotheses.

Some Helpful Tips, cont'd

- ✓ (7) **Divide into cases.** For example if the problem concerns integers, then you may want to divide it into two cases: one for even numbers and the other for odd numbers.
- ✓ (8) **Proof by contradiction.** If you make an assumption, and that assumption produces a statement that does not make sense, then you must conclude that your assumption is wrong. When you are stuck trying to justify some assertion, proof by contradiction is always a good thing to try.
- ✓ (9) **Transform/Restate** the problem, then try (1) (3) above.
- ✓ (10) **Working backward.** Start from the final goal (conclusion or desired form of an equation etc.) and seek what is necessary to reach the goal. Once it is found, it becomes new temporary goal. The process is to repeat until available data or familiar problem is reached.
- ✓ (11) **Simplify** the problem if possible. Take advantage of symmetries which often exist.

Shunichi Toida†:

- *"Keep in mind that first try may not work. But don't get discouraged. If one approach doesn't work, try another. You have to keep trying different approaches, different ideas. As you gain experience, your problem solving skills improve and you tend to find the right approach sooner."*

Professor Shunichi Toida is from Old Dominion University

†Some ideas and examples of the presented materials were inspired by Shunichi Toida's 2013 online class

Next class

- Topic: Propositional Logic
- Pre-class reading: Chap 1.1-1.2