

Discrete Mathematics – Toán Rời Rạc

Nguyễn Quân Bá Hồng*

Ngày 27 tháng 12 năm 2024

Tóm tắt nội dung

This text is a part of the series *Some Topics in Advanced STEM & Beyond*:

URL: https://nqbh.github.io/advanced_STEM/.

Latest version:

- *Discrete Mathematics – Toán Rời Rạc*.

PDF: URL: https://github.com/NQBH/advanced_STEM_beyond/blob/main/discrete_mathematics/NQBH_discrete_mathematics.pdf.

TEX: URL: https://github.com/NQBH/advanced_STEM_beyond/blob/main/discrete_mathematics/NQBH_discrete_mathematics.tex.

Mục lục

1 Basic Discrete Mathematics	1
2 Miscellaneous	7
3 Wikipedia's	7
3.1 Wikipedia/discrete mathematics	7
3.1.1 Topics	7
3.1.2 Challenges	9
3.2 Wikipedia/outline of discrete mathematics	9
3.2.1 Discrete mathematical disciplines	9
3.2.2 Concepts in discrete mathematics	9
3.2.3 Mathematicians associated with discrete mathematics	9
Tài liệu	9

1 Basic Discrete Mathematics

Resources – Tài nguyên.

1. [WR21]. RYAN T. WHITE, ARCHANA TIKAYAT RAY. *Practical Discrete Mathematics: Discover math principles that fuel algorithms for computer science & machine learning with Python*. [Amazon 44 ratings]

Amazon review. A practical guide simplifying discrete math for curious minds & demonstrating its application in solving problems related to software development, computer algorithms, & DS. Key Features:

- Apply math of countable objects to practical problems in computer science
- Explore modern Python libraries e.g. scikit-learn, NumPy, & SciPy for performing mathematics
- Learn complex statistical & mathematical concepts with help of hands-on examples & expert guidance

Book Description. Discrete mathematics deal with studying countable, distinct elements, & its principles are widely used in building algorithms for computer science & data science. Knowledge of discrete math concepts will help understand algorithms, binary, & general mathematics that sit at core of data-driven tasks.

Practical Discrete Mathematics is a comprehensive introduction for those who are new to mathematics of countable objects. This book will help you get up to speed with using discrete math principles to take your computer science skills to a more advanced level.

As learn language of discrete mathematics, also cover methods crucial to studying & describing computer science & ML objects & algorithms. Chaps will guide through how memory & CPUs work. In addition to this, understand how to analyze data for useful patterns, before finally exploring how to apply math concepts in network routing, web searching, & data science.

*A Scientist & Creative Artist Wannabe. E-mail: nguyenquanbahong@gmail.com. Bến Tre City, Việt Nam.

By end of this book, have a deeper understanding of discrete math & its applications in computer science, & be ready to work on real-world algorithm development & ML.

What You Will Learn.

- Understand terminology & methods in discrete math & their usage in algorithms & data problems
- use Boolean algebra in formal logic & elementary control structures
- Implement combinatorics to measure computational complexity & manage memory allocation
- Use random variables, calculate descriptive statistics, & find average-case computational complexity
- Solve graph problems involved in routing, pathfinding, & graph searches, e.g. depth-1st search
- Perform ML tasks e.g. data visualization, regression, & dimensionality reduction

Who this book is for. This book is for computer scientists looking to expand their knowledge of discrete math, core topic of their field. University students looking to get hands-on with computer science, mathematics, statistics, engineering, or related disciplines will also find this book useful. Basic Python programming skills & knowledge of elementary real-number algebra are required to get started with this book.

About the Authors. RYAN T. WHITE, Ph.D. is a mathematician, researcher, & consultant with expertise in ML & probability theory along with private-sector experience in algorithm development & data science. Dr. WHITE is an assistant professor of mathematics at Floria Institute of Technology, where he leads an active academic research program centered on stochastic analysis & related algorithms, heads private-sector projects in ML, participates in numerous scientific & engineering research projects, & teaches courses in ML, neural networks, probability, & statistics at undergraduate & graduate levels.

Archana Tikayat Ray s a Ph.D. student at Georgia Institute of Technology, Atlanta, where her research work is focused on ML & Natural Language Processing (NLP) applications. She has a master's degree from Georgia Tech as well, & a bachelor's degree in aerospace engineering from Florida Institute of Technology.

About the reviewer. VALERIY BABUSHKIN is senior director of data scientist at X5 Retail Group, where he leads a team of > 80 people in fields of ML, data analysis, computer vision, NLP, R&D, & A/B testing. VALERIY is a Kaggle competition grandmaster & an attending lecturer at National Research Institute's Higher School of Economics & Central Bank of Kazakhstan.

VALERIY served as a technical reviewer for books *AI Crash Course & Hands-On Reinforcement Learning with Python*, both published by Packt.

Preface. *Practical Discrete Mathematics* is a comprehensive introduction for those who are new to mathematics of countable objects. This book will help you get up to speed with using discrete math principles to take your computer science skills to another level. Learn language of discrete mathematics & methods crucial to studying & describing objects & algorithms from computer science & ML. Complete with real-world examples, this book covers internal workings of memory & CPUs, analyzes data for useful patterns, & shows how to solve problems in network routing, web searching, & data science.

Who this book is for. This book is for computer scientists looking to expand their knowledge of core of their field. University students seeking to gain expertise in computer science, mathematics, statistics, engineering, & related disciplines will also find this book useful. Knowledge of elementary real-number algebra & basic programming skills in any language are only requirements.

To get most out of this book. Knowledge of elementary real-number algebra & Python SPACE basic programming skills: main requirements for this book.

Will need to install Python – latest version, if possible – to run code in book. Will also need to install Python libraries listed in following table to run some of code in book. All code examples have been tested in JupyterLab using a Python 3.8 environment on Windows 10 OS, but they should work with any version of Python 3 in any OS compatible with it & with any modern integrated development environment, or simply a command line.

Python libraries: NumPy, matplotlib, pandas, scikit-learn, SciPy, seaborn. More information about installing Python & its libraries can be found in following links:

- Python: <https://www.python.org/downloads/>
- matplotlib: <https://matplotlib.org/3.3.3/users/installing.html>
- NumPy: <https://numpy.org/install/>
- pandas: https://pandas.pydata.org/pandas-docs/stable/getting_started/install.html
- scikit-learn: <https://scikit-learn.org/stable/install.html>
- SciPy: <https://www.scipy.org/install.html>
- seaborn: <https://seaborn.pydata.org/installing.html>

If use digital version of this book, advise to type code yourself or access code via GitHub repository. Doing so will help avoid any potential errors related to copying & pasting of code.

Download example code files. Can download example code files for this book from GitHub at <https://github.com/PacktPublishing/Practical-Discrete-Mathematics>. Also have other code bundles from rich catalog of books & videos available at <https://github.com/PacktPublishing/>.

Download color images. provide a PDF file that has color images of screenshots/diagrams used in this book. Can download via https://static.packt-cdn.com/downloads/9781838983147_ColorImages.pdf.

Part I: Basic Concepts of Discrete math.

- 1. Key Concepts, Notation, Set Theory, Relations, & Functions. an introduction to basic vocabulary, concepts, & notations of discrete mathematics.

This chap is a general introduction to main ideas of discrete mathematics. Alongside this, go through key terms & concepts in field. After that, cover set theory, essential notation & notations for referring to collections of mathematical object & combining or selecting them. Also think about mapping mathematical objects to 1 another with functions & relations & visualizing them with graphs. Topics covered in this chap:

- What is discrete mathematics?
- Elementary set theory
- Functions & relations

By end of chapter, should be able to speak in language of discrete mathematics & understand notation common to entire field.

- **What is discrete mathematics?** Discrete mathematics is study of countable, distinct, or separate mathematical structures. A good example is a pixel. From phones to computer monitors to televisions, modern screens are made up of millions of tiny dots called *pixels* lined up in grids. Each pixel lights up with a specified color on command from a device, but only a finite number of colors can be displayed in each pixel.

Millions of colored dots taken together form intricate patterns & give our eyes impression of shapes with smooth curves, as in boundary of following circle Fig. 1.1: **boundary of a circle**. But if zoom in & look closely enough, true “curves” are revealed to be jagged boundaries between differently colored regions of pixels, possibly with some intermediate colors: Fig. 1.2: **A zoomed-in view of circle**. Some other examples of objects studied in discrete mathematics are logical statements, integers, bits & bytes, graphs, trees, & networks. Like pixels, these too can form intricate patterns that we will try to discover & exploit for various purposes related to computer & data science throughout course of book.

In contrast, many areas of mathematics that may be more familiar, e.g. elementary algebra or calculus, focus on continuums. These are mathematical objects that take values over continuous ranges, e.g. set of numbers $x \in (0, 1)$, or mathematical functions plotted as smooth curves. These objects come with their own class of mathematical methods, but are mostly distinct from methods for discrete problems on which we will focus.

In recent decades, discrete mathematics has been a topic of extensive research due to advent of computers with high computational capabilities that operate in “discrete” steps & store data in “discrete” bits. This makes it important for us to understand principles of discrete mathematics as they are useful in understanding underlying ideas of software development, computer algorithms, programming languages, & cryptography. These computer implementations play a crucial role in applying principles of discrete mathematics to real-world problems.

Some real-world applications of discrete mathematics:

- * **Cryptography.** Art & science of converting data or information into an encoded form that can ideally only be decoded by an authorized entity. This field makes heavy use of number theory, study of counting numbers, & algorithms on base- n number systems. Will learn more about these topics in *Chap. 2: Formal Logic & Constructing Mathematical Proofs*.
- * **Logistics.** This field makes use of graph theory to simplify complex logistical problems by converting them to graphs. These graphs can further be used to find best routes for shipping goods & services, & so on. E.g., airlines use graph theory to map their global airplane routing & scheduling. Investigate some of these issues in Chaps of *Part II: Implementing Discrete Mathematics in Data & Computer Science*.
- * **Machine Learning.** Area that seeks to automate statistical & analytical methods so systems can find useful patterns in data, learn, & make decisions with minimal human intervention. Frequently applied to predictive modeling & web searches, see in *Chap 5: Elements of Discrete Probability*, & most of chaps in *Part III: Real-World Applications of Discrete Mathematics*.
- * **Analysis of Algorithms.** Any set of instructions to accomplish a task is an algorithm. An effective algorithm must solve problem, terminate in a useful amount of time, & not take up too much memory. To ensure 2nd condition, often necessary to count number of operations an algorithm must complete in order to terminate, which can be complex, but can be done through methods of combinatorics. 3rd condition requires a similar counting of memory usage. Encounter some of these ideas in *Chap. 4: Combinatorics Using SciPy*, *Chap. 6: Computational Algorithms in Linear Algebra*, & *Chap. 7: Computational Requirements for Algorithms*.
- * **Relational Databases.** They help to connect different traits between data fields. E.g., in a database containing information about accidents in a city, “relational feature” allows user to link location of accident to road condition, lighting condition, & other necessary information. A relational database makes use of concept of set theory in order to group together relevant information. See some of these ideas in *Chap. 8: Storage & Feature Extraction of Trees, Graphs, & Networks*.

Have a rough idea of what discrete mathematics is & some of its applications, discuss set theory, which forms basis for this field.

- o Elementary set theory.

“A set is a Many that allows itself to be thought of as a One.” – GEORG CANTOR

In mathematics, set theory is study of collections of objects, which is prerequisite knowledge for studying discrete mathematics. SKIP FAMILIARS

With knowledge of set theory, can now move on to learn about relations between different sets & functions, which help us to map each element from a set to exactly 1 element in another set.

- o Functions & relations.

“Gentlemen, mathematics is a language.” – JOSIAH WILLARD GIBBS

We are related to different people in different ways; e.g., relationship between a father & his son, relationship between a teacher & their students, & relationship between co-workers, to name just a few. Similarly, relationships exist between different elements in mathematics.

* **Example: Functions in elementary algebra.** Elementary algebra courses tend to focus on specific sorts of functions where domain & range are intervals of real number line. Domain values are usually denoted by x & values in range are denoted by y because set of ordered pairs (x, y) satisfying equation $y = f(x)$ plotted on Cartesian xy -plane form graph of function, as can be seen in Fig. 1.10: Cartesian xy -plane. While this typical type of functions may be familiar to most readers, concept of a function is more general than this. 1st, input or output is required to be a number. Domain of a function could consist of any set, so members of set may be points in space, graphs, matrices, arrays or strings, or any other types of elements.

In Python & most other programming languages, there are blocks of code known as “functions,” which programmers give names & will run when you call them. These Python functions may or may not take inputs (referred to as “parameters”) & return outputs, & each set of input parameters may or may not always return same output. As such, important: Python functions are not necessarily functions in mathematical sense, although some of them are.

An example of conflicting vocabulary in fields of mathematics & computer science. Next example will discuss some Python functions that are, & some that are not, functions in mathematical sense.

* **Example: Python functions vs. mathematical functions.** Consider `sort()` Python function, which is used for sorting lists. See this function applied to 2 lists – 1 list of numbers & 1 list of names:

```
numbers = [3, 1, 4, 12, 8, 5, 2, 9]
names = ['Wyatt', 'Brandon', 'Kumar', 'Eugene', 'Elise']
# Apply the sort() function to the lists
numbers.sort()
names.sort()
# Display the output
print(numbers)
print(names)
```

In each case, `sort()` function sorts list in ascending order by default (w.r.t. numerical order or alphabetical order). Furthermore, can say: `sort()` applies to any lists & is a function in mathematical sense. Indeed, it meets all criteria:

- (a) domain is all lists that can be sorted.
- (b) range is set of all such lists that have been sorted.
- (c) `sort()` always maps each list that can be inputted to a unique sorted list in range.

Consider now Python function random `shuffle()`, which takes a list as an input & puts it into a random order. Just like shuffle option on your favorite music app!

```
import random
# Set a random seed so the code is reproducible
random.seed(1)
# Run the random.shuffle() function 5 times and display the outputs
for i in range(0,5):
    numbers = [3, 1, 4, 12, 8, 5, 2, 9]
    random.shuffle(numbers)
    print(numbers)
```

This code runs a loop where each iteration sets list numbers to `[3,1,4,12,8,5,2,9]`, applies shuffle function to it, & prints output.

In each iteration, Python function `shuffle()` takes same input, but output is different each time. Therefore, Python function `shuffle()` is not a mathematical function. It is, however, a relation that can pair each list with any ordering of itself.

- o **Summary.** Discussed meaning of discrete mathematics & discrete objects. Furthermore, provided an overview of some of many applications of discrete mathematics in real world, especially in computer & data sciences, which will be discussed in depth in later chaps.

Have established some common language & notation of importance for discrete mathematics in form of set notation, which will allow us to refer to mathematical objects with ease, count size of sets, represent them as Venn diagrams, & much more. Beyond this, learned about a number of operations that allow us to manipulate sets by combining them, intersecting them, & finding complements. These give rise to some of foundational results in set theory in De Morgan's laws, which we will make use of in later chaps.

Took a look at ideas of functions & relations, which map mathematical objects e.g. numbers to 1 another. While certain types of functions may be familiar to reader from high school or secondary school, these familiar functions are typically defined on continuous domains. Since focus on discrete, rather than continuous, sets in discrete mathematics, drew distinction between familiar idea & a new one we need in this field. Similarly, showed difference between functions in mathematics & functions in Python & saw: some Python "functions" are mathematical functions, but others are not.

In remaining 4 chaps of *Part I: Core Concepts of Discrete Mathematics*, will fill our discrete mathematics toolbox with more tools, including logic in *Chap. 2: Formal Logic & Constructing Mathematical Proofs*, numerical systems, e.g. binary & decimal, in *Chap. 3: Computing with Base n Numbers*, counting complex sorts of objects, including permutations & combinations, in *Chap. 4: Combinatorics Using SciPy*, & dealing with uncertainty & randomness in *Chap. 5: Elements of Discrete Probability*. With this array of tools, will be able to consider more & more real-world applications of discrete mathematics.

- 2. **Formal Logic & Constructing Mathematical Proofs.** cover formal logic & binary & explain how to prove mathematical results.

This chap is an introduction to formal logic & mathematical proofs. 1st introduce some primary results of formal logic & prove logical statements with use of truth tables. In remainder of chap, consider most common methods of mathematical proofs (direct proof, proof by contradiction, & proof by mathematical induction) to build skills that you will need for more complex problems to come later. Cover topics:

- Formal logic & proofs by truth tables
- Direct mathematical proofs
- Proof by contradiction
- Proof by mathematical induction

By end of chap, will have a grasp of how formal logic provides a grounding for deductive thought, will have learned how to model logical problems with truth tables, will have proved claims with truth tables, & will have learned how to construct mathematical proofs using several methods: direct proof, proof by contradiction, & proof by mathematical induction. This short introduction to mathematical proofs will help you to learn how to think like a mathematician, use powerful deductive thought, & learn later material in book.

- Formal Logic & Proofs by Truth Tables.
- Direct Mathematical Proofs.
- Proof by Contradiction.
- Proof by Mathematical Induction.

◦ **Summary.** Introduced primary results of formal logic & proved logical statements by using truth tables. Also learned about constructing mathematical proofs using several methods, e.g. direct proofs, proofs by contradiction, & proofs by mathematical induction. In addition, these different methods for constructing mathematical proofs were accompanied by simple step-by-step examples to help you think like a mathematician & use deductive thought, which will be helpful for rest of chaps in this book. In next chap, learn about numbers in base n & perform some arithmetic operations with them. Will also learn about binary & hexadecimal numbers & their uses in computer science.

- 3. **Computing with Base- n Numbers.** discuss arithmetic in different numbering systems, including hexadecimal & binary.

We are all accustomed to decimal (base-10) numbers. Introduce numbers in other bases, describe arithmetic operations with those numbers, & convert numbers from 1 base to another. Then move to binary digits (base-2), which are foundation on which all computer operations are built, develop an approach to efficient arithmetic with them, & look at some of core uses of binary, including Boolean algebra. Lastly, discuss hexadecimal (base-16) numbers & their uses in computer science. Use Python code to do some computations e.g. converting decimal numbers to binary & hexadecimal & use Boolean operators to select & view data that satisfies a certain criterion. Cover following topics:

- Base- n numbers
- Converting between bases
- Binary numbers & their application
- Boolean algebra
- Hexadecimal numbers & their application

By end of this chap, should be able to write numbers in different bases & convert numbers from 1 base to another. E.g., 123 is a base-10 number that can be converted into other bases, depending on need. Also learn about importance of binary & hexadecimal number systems along with their applications in computer science.

- Understanding base- n numbers.
- Converting between bases.
- Binary numbers & their applications.

- Hexadecimal numbers & their application.
- Summary. Learned about numbers in different bases (decimal, binary, hexadecimal) & how can convert between bases. Binary numbers are a base-2 number system, whereas decimal numbers are base-10 & hexadecimal numbers are base-16, resp. Also learned about 1 very crucial application of binary number system – Boolean algebra & Boolean operators. Next chap, will learn about combinatorics, which includes study of permutations & combinations that will enable you to calculate amount of memory required to store certain kinds of data. Will learn about hashing & efficacy of brute force algorithms.
- 4. Combinatorics Using SciPy. explain how to count elements in certain types of discrete structures.
 - Fundamental counting rule.
 - Counting permutations & combinations of objects.
 - Applications to memory allocation.
 - Efficacy of brute-force algorithms.
 - Summary.
- 5. Elements of Discrete Probability. cover measuring chance & basics of Google's PageRank algorithm.
 - Basics of discrete probability.
 - Conditional probability & Bayes' theorem.
 - Bayesian spam filtering.
 - Random variables, means, & variance.
 - Google PageRank I.
 - Summary.

Part II: Implementing Discrete Mathematics in Data & Computer Science.

- 6. Computational Algorithms in Linear Algebra. explain how to solve algebra problems with Python using NumPy.
 - Understanding linear systems of equations.
 - Matrices & matrix representations of linear systems.
 - Solving small linear systems with Gaussian elimination.
 - Solving large linear systems with NumPy.
 - Summary.
- 7. Computational Requirements for Algorithms. give tools to determine how long algorithms take to run & how much space they require.
 - Computational complexity of algorithms.
 - Understanding Big-O Notation.
 - Complexity of algorithms with fundamental control structures.
 - Complexity of common search algorithms.
 - Common classes of computational complexity.
 - Summary.
- 8. Storage & Feature Extraction of Graphs, Trees, & Networks. cover storing graph structures & finding information about them with code.
 - Understanding graphs, trees, & networks.
 - Using graphs, trees, & networks.
 - Storage of graphs & networks.
 - Feature extraction of graphs.
 - Summary.
- 9. Searching Data Structures & Finding Shortest Paths. explain how to traverse graphs & figure out efficient paths between vertices.
 - Searching Graph & Tree data structures.
 - Depth-1st search (DFS).
 - Shortest path problem & variations of problem.
 - Finding Shortest Paths with Brute Force.
 - Dijkstra's Algorithm for Finding Shortest Paths.
 - Python Implementation of Dijkstra's Algorithm.
 - Summary.

Part III: Real-World Applications of Discrete Mathematics.

- 10. Regression Analysis with NumPy & Scikit-Learn. a discussion on prediction of variables in datasets containing multiple variables.

- Dataset.
- Best-fit lines & least-squares method.
- Least-squares lines with NumPy.
- Least-squares curves with NumPy & SciPy.
- Least-squares surfaces with NumPy & SciPy.
- Summary.
- 11. Web Searches with PageRank. show how to rank results of web searches to find most relevant web pages.
 - Development of Search Engines over time.
 - Google PageRank II.
 - Implementing PageRank algorithm in Python.
 - Applying Algorithm to Real Data.
 - Summary.
- 12. Principal Component Analysis with Scikit-Learn. explain how to reduce dimensionality of high-dimensional datasets to save space & speed up ML.
 - Understanding eigenvalues, eigenvectors, & orthogonal bases.
 - Principal component analysis approach to dimensionality reduction.
 - Scikit-learn implementation of PCA.
 - An application to real-world data.
 - Summary.

2 Miscellaneous

3 Wikipedia's

3.1 Wikipedia/discrete mathematics

“*Discrete mathematics* is study of **mathematical structures** that can be considered “discrete” (in a way analogous to **discrete variables**, having a **bijection** with \mathbb{N}) rather than “continuous” (analogously to **continuous functions**). Objects studied in discrete mathematics include integers, **graphs**, & **statements** in **logic**. By contrast, discrete mathematics excludes topics in “continuous mathematics” e.g. real numbers, calculus or **Euclidean geometry**. Discrete objects can often be **enumerated** by integers; more formally, discrete mathematics has been characterized as branch of mathematics dealing with **countable sets** (finite sets or sets with same **cardinality** as \mathbb{N}). However, there is no exact definition of term “discrete mathematics”.

Set of objects studied in discrete mathematics can be finite or infinite. Term *finite mathematics* is sometimes applied to parts of field of discrete mathematics that deals with finite sets, particularly those areas relevant to business.

Graphs e.g. these are among objects studied by discrete mathematics, for their interesting **mathematical properties**, their usefulness as models of real-world problems, & their importance in developing computer algorithms.

Research in discrete mathematics increased in latter half of 20th century partly due to development of **digital computers** which operate in “discrete” steps & store data in “discrete” bits. Concepts & notations from discrete mathematics are useful in studying & describing objects & problems in branches of computer science, e.g. **computer algorithms**, **programming languages**, **cryptography**, **automated theorem proving**, & **software development**. Conversely, computer implementations are significant in applying ideas from discrete mathematics to real-world problems.

Although main objects of study in discrete mathematics are discrete objects, **analytic** methods from “continuous” mathematics are often employed as well.

In university curricula, discrete mathematics are discrete objects, **analytic** methods from “continuous” mathematics are often employed as well.

In university curricula, discrete mathematics appeared in 1980s, initially as a computer science support course; its contents were somewhat haphazard at time. Curriculum has thereafter developed in conjunction with efforts by **ACM** & **MAA** into a course that is basically intended to develop **mathematical maturity** in 1st-year students; therefore, it is nowadays a prerequisite for mathematics majors in some universities as well. Some high-school-level discrete mathematics textbooks have appeared as well. At this level, discrete mathematics is sometimes seen as a preparatory course, like **precalculus** in this respect.

Fulkerson Prize is awarded for outstanding papers in discrete mathematics.

3.1.1 Topics

1. Theoretical computer science. **Complexity** studies time taken by algorithms, e.g. this **quick sort**. **Theoretical computer science** includes areas of discrete mathematics relevant to computing. It draws heavily on **graph theory** & **mathematical logic**. Included within theoretical computer science is study of algorithms & data structures. **Computability** studies what can be computed in principle, & has close ties to logic, while complexity studies time, space, & other resources taken by computations. **Automata theory** & **formal language** theory are closely related to computability. **Petri nets** & **process algebras** are used to model computer systems, & methods from discrete mathematics are used in analyzing **VLSI** electronic circuits.

Computational geometry applies computer algorithms to representations of geometrical objects. **Computational geometry** applies algorithms to geometrical problems & representations of geometrical objects, while **computer image analysis** applies them to representations of images. Theoretical computer science also includes study of various continuous computational topics.

2. Information theory. **ASCII** codes for word “Wikipedia”, given here in **binary**, provide a way of representing word in **information theory**, as well as for information-processing algorithms. **Information theory** involves quantification of **information**. Closely related is **coding theory** which is used to design efficient & reliable data transmission & storage methods. Information theory also includes continuous topics e.g.: **analog signals**, **analog coding**, **analog encryption**.

3. Logic. **Mathematical logic** is study of principles of valid reasoning & **inference**, as well as of **consistency**, **soundness**, & **completeness**. E.g., in most systems of logic (but not in **intuitionistic logic**) **Peirce’s law** $((P \rightarrow Q) \rightarrow P) \rightarrow P$ is a theorem. For classical logic, it can be easily verified with a **truth table**. Study of **mathematical proof** is particularly important in logic, & has accumulated to **automated theorem proving** & **formal verification** of software.

Logical formulas are discrete structures, as are **proofs**, which form finite **trees** or, more generally, **directed acyclic graph** structures (with each **inference step** combining 1 or more **premise** branches to give a single conclusion). **Truth values** of logical formulas usually form a finite set, generally restricted to 2 values: true & false, but logic can also be continuous-valued, e.g., **fuzzy logic**. Concepts e.g. infinite proof trees or infinite derivation trees have also been studied, e.g., **infinitary logic**.

4. Set theory. **Set theory** is branch of mathematics that studies **sets**, which are collections of objects, e.g. {blue, white, red} or (infinite) set of all **prime numbers**. **Partially ordered sets** & sets with other **relations** have applications in several areas.

In discrete mathematics, **countable sets** (including **finite sets**) are main focus. Beginning of set theory as a branch of mathematics is usually marked by **GEORGE CANTOR**’s work distinguishing between different kinds of **infinite set**, motivated by study of trigonometric series, & further development of theory of infinite sets is outside scope of discrete mathematics. Indeed, contemporary work in **descriptive set theory** makes extensive use of traditional continuous mathematics.

5. Combinatorics. **Combinatorics** studies ways in which discrete structures can be combined or arranged. **Enumerative combinatorics** concentrates on counting number of certain combinatorial objects – e.g., **12fold way** provides a unified framework for counting **permutations**, **combinations**, & **partitions**. **Analytic combinatorics** concerns enumeration (i.e., determining number) of combinatorial structures using tools from **complex analysis** & probability theory. In contrast with enumerative combinatorics which uses explicit combinatorial formulae & **generating functions** to describe results, analytic combinatorics aims at obtaining **asymptotic formulae**. **Topological combinatorics** concerns use of techniques from **topology** & **algebraic topology/combinatorial topology** in **combinatorics**. Design theory is a study of **combinatorial designs**, which are collections of subsets with certain intersection properties. **Partition theory** studies various enumeration & asymptotic problems related to **integer partitions**, & is closely related to **q-series**, **special functions**, & **orthogonal polynomials**. Originally a part of number theory & analysis, partition theory is now considered a part of combinatorics or an independent field. **Order theory** is study of **partially ordered sets**, both finite & infinite.

6. Graph theory. **Graph theory** has close links to **group theory**. This **truncated tetrahedron** graph is related to **alternating group** A_4 . **Graph theory**, study of **graphs** & **networks**, is often considered part of combinatorics, but has grown large enough & distinct enough, with its own kind of problems, to be regarded as a subject in its own right. Graphs are 1 of prime objects of study in discrete mathematics. They are among most ubiquitous models of both natural & human-made structures. They can model many types of relations & process dynamics in physical, biological & social systems. In computer science, they can represent networks of communication, data organization, computational devices, flow of computation, etc. In mathematics, they are useful in geometry & certain parts of topology, e.g. **knot theory**. **Algebraic graph theory** has close links with group theory & **topological graph theory** has close links to topology. There are also **continuous graphs**; however, for most part, research in graph theory falls within domain of discrete mathematics.

7. Number theory. **Ulam spiral** of numbers, with black pixels showing prime numbers. This diagram hints at patterns in **distribution** of prime numbers. **Number theory** is concerned with properties of numbers in general, particularly integers. It has applications to **cryptography** & **cryptanalysis**, particularly with regard to **modular arithmetic**, **diophantine equations**, linear & quadratic congruences, prime numbers & **primality testing**. Other discrete aspects of number theory include **geometry of numbers**. In **analytic number theory**, techniques from continuous mathematics are also used. Topics that go beyond discrete objects include **transcendental numbers**, **diophantine approximation**, **p-adic analysis** & **function fields**.

8. Algebraic structures. Main article: **Wikipedia/abstract algebra**. **Algebraic structures** occur as both discrete examples & continuous examples. Discrete algebras include: **Boolean algebra** used in **logic gates** & programming; **relational algebra** used in **databases**; discrete & finite versions of groups, rings, & fields are important in **algebraic coding theory**; discrete **semigroups** & **monoids** appear in theory of **formal languages**.

9. Discrete analogues of continuous mathematics. There are many concepts & theories in continuous mathematics which have discrete versions, e.g. **discrete calculus**, **discrete Fourier transforms**, **discrete geometry**, **discrete logarithms**, **discrete differential geometry**, **discrete exterior calculus**, **discrete Morse theory**, **discrete optimization**, **discrete probability theory**, **discrete probability distribution**, **difference equations**, **discrete dynamical systems**, & **discrete vector measures**.

- Calculus of finite differences, discrete analysis. In **discrete calculus** & **calculus of finite differences**, a function defined on an interval of integers is usually called a **sequence**. A sequence could be a finite sequence from a data source or an infinite

sequence from a **discrete dynamical system**. Such a discrete function could be defined explicitly by a list (if its domain is finite), or by a formula for its general term, or it could be given implicitly by a **recurrence relation** or **difference equation**. Difference equations are similar to **differential equations**, but replace **differentiation** by taking difference between adjacent terms; they can be used to approximate differential equations or (more often) studied in their own right. Many questions & methods concerning differential equations have counterparts for difference equations. E.g., where there are **integral transforms** in **harmonic analysis** for studying continuous functions for analogue signals, there are **discrete transforms** for discrete functions or digital signals. As well as **discrete metric spaces**, there are more general **discrete topological spaces**, **finite metric spaces**, **finite topological spaces**.

Time scale calculus is a unification of theory of **difference equations** with that of **differential equations**, which has applications to fields requiring simultaneous modeling of discrete & continuous data. Another way of modeling such a situation is notion of **hybrid dynamical systems**.

- Discrete geometry. **Discrete geometry** & combinatorial geometry are about combinatorial properties of *discrete collections* of geometrical objects. A long-standing topic in discrete geometry is **tiling of plane**.

In **algebraic geometry**, concept of a curve can be extended to discrete geometries by taking **spectra** of **polynomials rings** over **finite fields** to be models of **affine spaces** over that field, & letting **subvarieties** or spectra of other rings provide curves that lie in that space. Although space in which curves appear has a finite number of points, curves are not so much sets of points as analogues of curves in continuous settings. E.g., every point of form $V(x - c) \subset \text{Spec}K[x] = \mathbb{A}^1$ for K a field can be studied either as $\text{Spec}K[x]/(x - c) \cong \text{Spec}K$, a point, or as spectrum $\text{Spec}K[x]_{(x-c)}$ of **local ring at $(x - c)$** , a point together with a neighborhood around it. Algebraic varieties also have a well-defined notion of **tangent space** called **Zariski tangent space**, making many features of calculus applicable even in finite settings.

- Discrete modeling. In **applied mathematics**, **discrete modeling** is discrete analogue of **continuous modeling**. In discrete modeling, discrete formulae are fit to **data**. A common method in this form of modeling is to use **recurrence relation**. **Discretization** concerns process of transferring continuous models & equations into discrete counterparts, often for purposes of making calculations easier by using approximations. **Numerical analysis** provides an important example.

3.1.2 Challenges

Much research in **graph theory** was motivated by attempts to prove: all maps can be **colored** using **only 4 colors** so that no areas of same color share an edge. **KENNETH APPEL** & **WOLFGANG HAKEN** proved this in 1976.

History of discrete mathematics has involved a number of challenging problems which have focused attention within areas of field. In graph theory, much research was motivated by attempts to prove **4 color theorem**, 1st stated in 1852, but not proved until 1976 (by **KENNETH APPEL** & **WOLFGANG HAKEN**, using substantial computer assistance).

In logic, **2nd problem** on **DAVID HILBERT**'s list of open **problems** presented in 1900 was to prove: axioms of arithmetic are consistent. **Gödel's 2nd incompleteness theorem**, proved in 1931, showed: this was not possible – at least not within arithmetic itself. **Hilbert's 10th problem** was to determine whether a given polynomial **Diophantine equation** with integer coefficients has an integer solution. In 1970, **YURI MATIYASEVICH** proved: this **could not be done**.

Need to **break** German codes in **World War II** led to advances in **cryptography** & **theoretical computer science**, with **1st programmable digital electronic computer** being developed at England's **Bletchley Park** with guidance of **ALAN TURING** & his seminal work, *On Computable Numbers*. **Cold War** meant: cryptography remained important, with fundamental advances e.g. **public-key cryptography** being developed in following decades. **Telecommunication industry** has also motivated advances in discrete mathematics, particularly in graph theory & **information theory**. **Formal verification** of statements in logic has been necessary for **software development** of **safety-critical systems**, & advances in **automated theorem proving** have been driven by this need.

Computational geometry has been an important part of **computer graphics** incorporated into modern **video games** & **computer-aided design** tools.

Several fields of discrete mathematics, particularly theoretical computer science, graph theory, & **combinatorics**, are important in addressing challenging **bioinformatics** problems associated with understanding **tree of life**.

Currently, 1 of most famous open problems in theoretical science is **P = NP problem**, which involves relationship between **complexity classes P** & **NP**. **Clay Mathematics Institute** has offered a \$1 million USD prize for 1st correct proof, along with prizes for **6 other mathematical problems**.” – [Wikipedia/discrete mathematics](https://en.wikipedia.org/wiki/Discrete_mathematics)

3.2 Wikipedia/outline of discrete mathematics

“**Discrete** is study of **mathematical structures** that are fundamentally **discrete** rather than **continuous**. In contrast to real numbers that have property of varying “smoothly”, objects studied in discrete mathematics – e.g. integers, graphs, & statements in logic – do not vary smoothly in this way, but have distinct, separated values. Discrete mathematics, therefore, excludes topics in “continuous mathematics” e.g. calculus & analysis.

Included below are many of standard term used routinely in university-level courses & in research papers. This is not, however, intended as a complete list of mathematical terms; just a selection of typical **terms of art** that may be encountered.

- **Logic**: Study of correct reasoning.
- Modal logic – Type of formal logic
- Set theory – Branch of mathematics that studies sets

3.2.1 Discrete mathematical disciplines

3.2.2 Concepts in discrete mathematics

3.2.3 Mathematicians associated with discrete mathematics

” – [Wikipedia/outline of discrete mathematics](#)

Tài liệu

[WR21] Ryan T White and Archana Tikayat Ray. *Practical Discrete Mathematics: Discover math principles that fuel algorithms for computer science & machine learning with Python*. Packt Publishing, 2021, p. 330.